z/OS

Integrated Security Services LDAP Server Administration and Use

SC24-5923-06

z/OS

Integrated Security Services LDAP Server Administration and Use

SC24-5923-06

Note

Before using this information and the product it supports, be sure to read the general information under “Appendix G. Notices”.

Acknowledgements

Some of the material contained in this document is a derivative of LDAP documentation provided with the University of Michigan LDAP reference implementation (Version 3.3). Copyright © 1992-1996, Regents of the University of Michigan, All Rights Reserved.

This product includes software developed by the University of California, Berkeley and its contributors.

This product includes software developed by NEC Systems Laboratory.

Seventh Edition, March 2005

This edition, SC24-5923-06, applies to Version 1 Release 6 of z/OS (program number 5694-A01) and Version 1 Release 6 of z/OS.e (program number 5655-G52), and to all subsequent releases until otherwise indicated in new editions.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this document, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9405
FAX (Other Countries):
Your International Access Code +1+845+432-9405

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrcfs@us.ibm.com
World Wide Web: http://www.ibm.com/servers/eserver/zseries/zos/webqs.html

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this document
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

About this document

This document supports z/OS (5694-A01) and z/OS.e (5655-G52) and explains the LDAP server. The LDAP server supports Lightweight Directory Access Protocol (LDAP) and runs as a stand-alone daemon. It is based on a client/server model that provides client access to an LDAP server. The LDAP server provides an easy way to maintain directory information in a central location for storage, updating, retrieval, and exchange.

Intended audience

This document is intended to assist system administrators. System administrators should be experienced and have previous knowledge of directory services. It is also intended for anyone that will be implementing the directory service.

How this document is organized

This document is divided into the following parts:

- Part 1, “Administration,” on page 1
- Part 2, “Use,” on page 159
- Part 3, “Messages,” on page 375
- Part 4, Appendixes, Appendix A, “Minimum schema for TDBM and GDBM,” on page 439

Conventions used in this document

This document uses the following typographic conventions:

Bold **Bold** words or characters represent API names, attributes, status codes, environment variables, parameter values, and system elements that you must enter into the system literally, such commands, options, or path names.

Italic *Italic* words or characters represent values for variables that you must supply.

`Example Font`
Examples and information displayed by the system appear in `constant width type style`.

[ ] Brackets enclose optional items in format and syntax descriptions.

{ } Braces enclose a list from which you must choose an item in format and syntax descriptions.

| A vertical bar separates items in a list of choices.

< > Angle brackets enclose the name of a key on the keyboard.

... Horizontal ellipsis points indicate that you may repeat the preceding item one or more times.

**** A backslash is used as a continuation character when entering commands from the shell that exceed one line (255 characters). If the command exceeds one line, use the backslash character \ as the last nonblank character on the line to be continued, and continue the command on the next line.

Where to find more information

Where necessary, this document references information in other documents. For complete titles and order numbers of the documents for all products that are part of z/OS, refer to *z/OS Information Roadmap*.

For a list of titles and order numbers of the documents that are useful for z/OS LDAP, see Bibliography.

Softcopy publications

The z/OS LDAP library is available on a CD-ROM, z/OS Collection, SK3T-4269. The CD-ROM online library collection is a set of unlicensed documents for z/OS and related products that includes the IBM Softcopy Reader, a program that enables you to view the BookManager files. This CD-ROM also contains the Portable Document Format (PDF) files. You can view or print these files with the Adobe Reader.

z/OS online library

The softcopy z/OS publications are also available for web browsing and for viewing or printing PDFs using the following URL:

`www.ibm.com/servers/eserver/zseries/zos/bkserv/e0zlib`

You can also provide comments about this document and any other z/OS documentation by visiting that URL. Your feedback is important in helping to provide the most accurate and high-quality information.

Accessing licensed documents on the Web

z/OS licensed documentation in PDF format is available on the Internet at the IBM Resource Link Web site:

`www.ibm.com/servers/resourcelink`

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID, password, and z/OS licensed book key code. The z/OS order you received provides a memo that includes your key code.

To obtain your IBM Resource Link user ID and password, logon to:

`www.ibm.com/servers/resourcelink`

To register for access to the z/OS licensed books:

1. Logon to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.
3. Select **Access Profile**.
4. Select **Request Access to Licensed books**.
5. Supply your key code where requested and select the **Submit** button.

If you supplied the correct key code you will receive confirmation that your request is being processed.

After your request is processed you will receive an e-mail confirmation.

Note: You cannot access the z/OS licensed books unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

To access the licensed books:

1. Log on to Resource Link using your Resource Link user ID and password.
2. Select **Library**.
3. Select **zSeries**.
4. Select **Software**.
5. Select **z/OS**.
6. Access the licensed book by selecting the appropriate element.

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM® messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from the following locations to find IBM message explanations for z/OS® elements and features, z/VM®, VSE/ESA™, and Clusters for AIX® and Linux:

- The Internet. You can access IBM message explanations directly from the LookAt Web site at http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/.
- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e systems to access IBM message explanations, using LookAt from a TSO/E command line (for example, TSO/E prompt, ISPF, or z/OS UNIX® System Services running OMVS).
- Your Microsoft® Windows® workstation. You can install code to access IBM message explanations on the *z/OS Collection* (SK3T-4269), using LookAt from a Microsoft Windows command prompt (also known as the DOS command line).
- Your wireless handheld device. You can use the LookAt Mobile Edition with a handheld device that has wireless access and an Internet browser (for example, Internet Explorer for Pocket PCs, Blazer, or Eudora for Palm OS, or Opera for Linux handheld devices). Link to the LookAt Mobile Edition from the LookAt Web site.

You can obtain code to install LookAt on your host system or Microsoft Windows workstation from a disk on your *z/OS Collection* (SK3T-4269), or from the LookAt Web site (click **Download**, and select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

Summary of Changes

Summary of changes
for SC24-5923-06
z/OS Version 1 Release 6

This document contains information previously presented in *z/OS Security Server LDAP Server Administration and Use*, SC24-5923-05, which supports z/OS Version 1 Release 6.

The following summarizes the changes to that information:

New information

- RACF attributes RSLKEY and TSLKEY have been added.
- The following messages have been added:
  GLD4021E — GLD4025E
- Large group support has been enhanced to increase the number of members that can be contained within a single access group.

Changed information:

- Concurrency considerations between Modify DN operations and other LDAP operations has been changed.

Summary of changes
for SC24-5923-05
z/OS Version 1 Release 6

This document contains information previously presented in *z/OS Security Server LDAP Server Administration and Use*, SC24-5923-03 and SC24-5923-04, which supports z/OS Version 1 Release 4.

The following summarizes the changes to that information:

New information

- New attributes have been added to the RACF user OMVS segment: **MEMLIMIT** and **SHMEMMAX**.
- New attributes have been added to the rootDSE entry to publish the capabilities of the LDAP server: **ibm-supportedCapabilities** and **ibm-enabledCapabilities**
- Support has been added for creating alias entries and dereferencing them during search.
- Change log support has been added for changes to TDBM entries and to RACF objects. This includes a new backend section (GDBM) in the configuration file.
- The following extended operation has been added: **changeLogAddEntryRequest**.
- Support has been added for retrieving a RACF password envelope.
- Support for IPv6 addressing has been added.
- Entry and filter caching have been added to improve search performance.
- New performance tuning chapter including information on tuning DB2, controlling caching via new configuration options, tuning thread usage, and using the enhanced monitoring has been added.
- New debug chapter containing information on common configuration and run-time problems has been added.
- Enhanced updating of an existing schema, including a new configuration option and a new server control, **schemaReplaceByValueControl**.
- Support for enhanced static, dynamic, and nested groups of users has been added. This support allows for these group definitions to be added as **aclentry** attributes on entries within the TDBM and GDBM backends.

- Support has been added to enable **ibm-allmembers** and **ibm-allgroups** operational attribute search and comparison operations.
- DB2 and TCPIP failure recovery information has been added.
- Support has been added for peer replication.
- Information has added for a new server control, **PersisentSearch**.
- The following configuration file options have been added: **aclSourceCacheSize**, **dnCacheSize**, **dnToEidCacheSize**, **entryCacheSize**, **entryOwnerCacheSize**, **filterCacheBypassLimit**, **filterCacheSize**, **changeLogging**, **changeLoggingParticipant** , **changeLogMaxEntries**, **changelogMaxAge**, **db2terminate**, **peerServerDN**, **peerServerPW**, **persistentSearch**, and **schemaReplaceByValue**.
- The following reason codes have been added:
  - R000012 - R000015
  - R001073
  - R002019 - R002026
  - R003127
  - R004153 - R004174
  - R006016 - R006022
  - R007040 - R007052
- The following messages have been added:
  - GLD0241A — GLD0266A
  - GLD3144A — GLD3153A
  - GLD7013A

Changed information

- The Migration chapter has been updated and the Summary of Interface section has been removed. See *z/OS Migration, GA22-7499* for complete migration information. Also, see the *z/OS Summary of Message and Interface Changes, SA22-7505* for information on interface changes.
- The Running and using the LDAP entry UUID utility and Running and using the password encryption utility chapters have been merged into one chapter, Chapter 11, “Running and using the LDAP backend utilities,” on page 135.
- **ldapcnf** information has been updated to support configuring the change log and to indicate that configuring TDBM is no longer required.
- The instructions for migrating to new levels of TDBM schema have been changed.
- The following configuration file options have been updated: **altserver**, **listen**, **masterServer**, and **referral**.
- The following messages have been updated:
  - GLD0142E
  - GLD0226E
  - GLD0227E
  - GLD3009I
  - GLD3010I
  - GLD3084A
  - GLD3085A
  - GLD3120A
  - GLD3133A
  - GLD3136I

Deleted information:

- Appendix, LDAP server configuration file (slapd.conf), has been removed. See the /usr/lpp/ldap/ldap/etc/slapd.conf file.
- Appendix, Sample JCL, has been removed. See members LDAPSRV, LDF2TDBM, TDBM2LDF, LDAPUUID, DB2PWDEN in *GLDHLQ*.SGLDSAMP dataset.
- Appendix, Sample LDIF input file, has been removed. See /usr/lpp/ldap/examples/sample_server/sample.ldif file.
- The /usr/lpp/ldap/etc/dbSpufi.spufi file is no longer shipped. It has been replaced by two files, tdbSpufi.spufi and gdbSpufi.spufi.
- The /usr/lpp/ldap/examples/sample_server/tdbm_samp_db.spufi and tdbm_samp_index.spufi files are no longer used by the sample server and have been removed.
- The following schema files in the /usr/lpp/ldap/etc directory are no longer shipped. Instead, their contents have been merged into the schema.user.ldif and schema.IBM.ldif schema files:

  ChangeLog.ldif, CommServer.ldif, ComponentBroker.ldif, DB2.ldif, DMTF.ldif, EIM.ldif, EntrustPKIV4.ldif, EntrustPKIV5.ldif, IBM.ldif, Kerberos-V1.ldif, Kerberos-V3.ldif, ManagedSystemInfrastructure.ldif, MCI.ldif, MetaDirectory.ldif, MS.ActiveDirectory.ldif, NativeAuthentication.ldif, Netscape-V2.ldif, Netscape.ldif, NFI.ldif, nisSchema.ldif, OnDemandServer.ldif, OtherStandard.ldif, PolicyDirector.ldif, RACF.2.ldif, RACF.ldif, RegisteredSoftware.ldif, RFC2252.ldif, RFC2256.ldif, RFC2587.ldif, RFC2713.ldif, RFC2714.ldif, SecurityIdentities.ldif, System-V2.ldif, System.ldif, UniversalMessaging.ldif, UNIX.ldif, WebSphereNaming.ldif, X.520.ldif
- This document has been enabled for the following z/OS library center advanced searches: *Reference*, *examples*, *tasks*, and *concepts*.

Summary of changes
for SC24-5923-04
as updated December, 2002

Changed information

- Configuration option **idleConnectionTimeout**, has been updated.
- This book has been LOOKAT enabled.

Summary of changes
for SC24-5923-03
z/OS Version 1 Release 4

This document contains information previously presented in *z/OS Security Server LDAP Server Administration and Use*, SC24-5923-02, which supports z/OS Version 1 Release 2 and z/OS Version 1 Release 3.

The following summarizes the changes to that information:

New information

- CRAM-MD5 (RFC 2195) and DIGEST-MD5 (RFC 2831) authentication
- New global CRAM-MD5 and DIGEST-MD5 configuration option: **digestRealm**
- LDAP IBM-entryuuid support
- New LDAP entry UUID configuration option: **serverEtherAddr**
- New LDAP entry UUID utility: **ldapadduuids**
- The LDAP server supports both Secure Sockets Layer (SSL) Version 3 and Transport Layer Security (TLS) Version 1 for secure protected sessions. Where the book discusses both SSL and TLS together, they are written as SSL/TLS. Support for client connections that begin on the non-secure port which then switch to a secure (SSL/TLS) protected session using the LDAP Start TLS extended operation. This support implements RFC 2830.

- ACL enhancements to allow attribute-level access control and the ability to explicitly deny access to information.
- Full support for Modify DN operations, including subtree relocation
- New supported server controls: **IBMModifyDNTimelimitControl** and **IBMModifyDNRealignDNAttributesControl**
- Server activity logging
- New activity logging configuration option: **logfile**
- Monitor support
- Migration information
- New LDAP server messages and updated severity levels - new messages:
  - GLD0207I - GLD0241A
  - GLD2100A - GLD2125A
  - GLD3133A - GLD3143I
  - GLD4013A - GLD4020I
  - GLD5006A
- Information is added to indicate this document supports z/OS.e

Changed information

- The **db2pwden** utility requires that the value specified with the **-N** operand be the certificate label. The certificate name for this operand is no longer supported.
- The following configuration options are now ignored by the LDAP server: **attribute**, **index**, **objectclass**, **verifySchema**, **tbspaceentry**, **tbspacemutex**, **tbspace32k**, and **tbspace4k**.
- With the removal of the RDBM backend, **rdbm** can no longer be specified as the *dbtype* on the **database** configuration option.
- The **database** configuration option has been extended to allow specification of a unique name for each configured backend instance.
- The **slapd.conf** file has been updated.
- Severity indicators have changed for most server messages.
- The instructions for migrating to new levels of TDBM schema have been changed.

Deleted information

- RDBM backend (including the **ldif2db** and **db2ldif** utilities and the **ldapcp** command)
- Online documentation in HTML form for the z/OS LDAP client APIs. Prior to z/OS V1R4, the LDAP Programming guide documentation was shipped in HTML form as part of the LDAP product. This documentation is no longer shipped. Refer to “z/OS online library” on page xviii for information about accessing online copies of this documentation.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Part 1. Administration

Chapter 1. Introducing the z/OS LDAP server

The z/OS Lightweight Directory Access Protocol (LDAP) server, part of the Integrated Security Services for z/OS, is based on a client/server model that provides client access to an LDAP server. An LDAP directory provides an easy way to maintain directory information in a central location for storage, update, retrieval, and exchange.

The LDAP server provides the following functions:
- Interoperability with other LDAP clients
- Access controls on directory information, using static, dynamic, and nested groups
- Secure Sockets Layer (SSL) communication (SSL V3 and TLS V1)
- Start TLS (Transport Layer Security) activation of secure communication
- Client and server authentication using SSL/TLS
- Password encryption
- Replication
- Referrals
- Aliases
- Change logging
- LDAP Version 2 and Version 3 protocol support
- Schema publication and update
- Kerberos authentication
- Native authentication
- CRAM-MD5 (Challenge-Response Authentication Method) and DIGEST-MD5 authentication
- Root DSE information
- LDAP access to information stored in RACF®
- Use of DB2® data sharing in sysplex configurations

This book describes how to install, configure, and run the stand-alone LDAP server and other LDAP programs. It is intended for newcomers and experienced administrators alike. This section provides a basic introduction to directory services, and the directory service provided by the LDAP server in particular.

z/OS Integrated Security Services LDAP Client Programming describes the LDAP client application programming interfaces (APIs) you can use to develop LDAP applications.

What is a directory service?

A directory is like a database, but tends to contain more descriptive, attribute-based information. The information in a directory is generally read much more often than it is written. As a consequence, directories do not usually implement the complicated transaction or rollback schemes that relational databases use for doing high-volume complex updates. Directory updates are typically simple all-or-nothing changes, if they are allowed at all. Directories are tuned to give quick-response to high-volume lookup or search operations. They may have the ability to replicate information widely in order to increase availability and reliability, while reducing response time. When directory information is replicated, temporary inconsistencies between the replicas are considered acceptable, as long as they get in sync eventually.

There are many different ways to provide a directory service. Different methods allow different kinds of information to be stored in the directory, place different requirements on how that information can be referenced, queried and updated, how it is protected from unauthorized access, and so on. Some directory services are local, providing service to a restricted context (for example, the finger service on a single machine). Other services are global, providing service to a much broader context (for example, the entire Internet). Global services are usually distributed, meaning that the data they contain is spread across many machines, all of which cooperate to provide the directory service. Typically a global service defines a uniform namespace which gives the same view of the data no matter where you are in relation to the data itself.

What is LDAP?

The LDAP server's model for the directory service is based on a global directory model called LDAP, which stands for the Lightweight Directory Access Protocol. LDAP Version 2 (V2) and LDAP Version 3 (V3), both supported in z/OS, are directory service protocols that run over TCP/IP. The details of LDAP V2 are defined in Internet Engineering Task Force (IETF) Request for Comments (RFC) 1777 *The Lightweight Directory Access Protocol* and the details of LDAP V3 are defined in the set of IETF RFCs 2251 - 2256. "RFCs supported by z/OS LDAP" on page 10 shows the entire list of supported RFCs.

This section gives an overview of LDAP from a user's perspective.

What kind of information can be stored in the directory?

The LDAP directory service model is based on *entries*. An entry is a collection of attributes that has a name, called a *distinguished name* (DN). The DN is used to refer to the entry unambiguously. Each of the entry's attributes has a type and one or more values. The types are typically mnemonic strings, like **cn** for common name, or **mail** for e-mail address. The values depend on what type of attribute it is. For example, a **mail** attribute might contain an e-mail address with an attribute value of `thj@vnet.ibm.com`. A **jpegPhoto** attribute would contain a photograph in binary JPEG format.

How is the information arranged?

In LDAP, directory entries are arranged in a hierarchical tree-like structure that sometimes reflects political, geographic or organizational boundaries. Entries representing countries appear at the top of the tree. Below them are entries representing states or national organizations. Below them might be entries representing people, organizational units, printers, documents, or just about anything else you can think of. Figure 1 on page 5 shows an example LDAP directory tree, which should help make things clear.

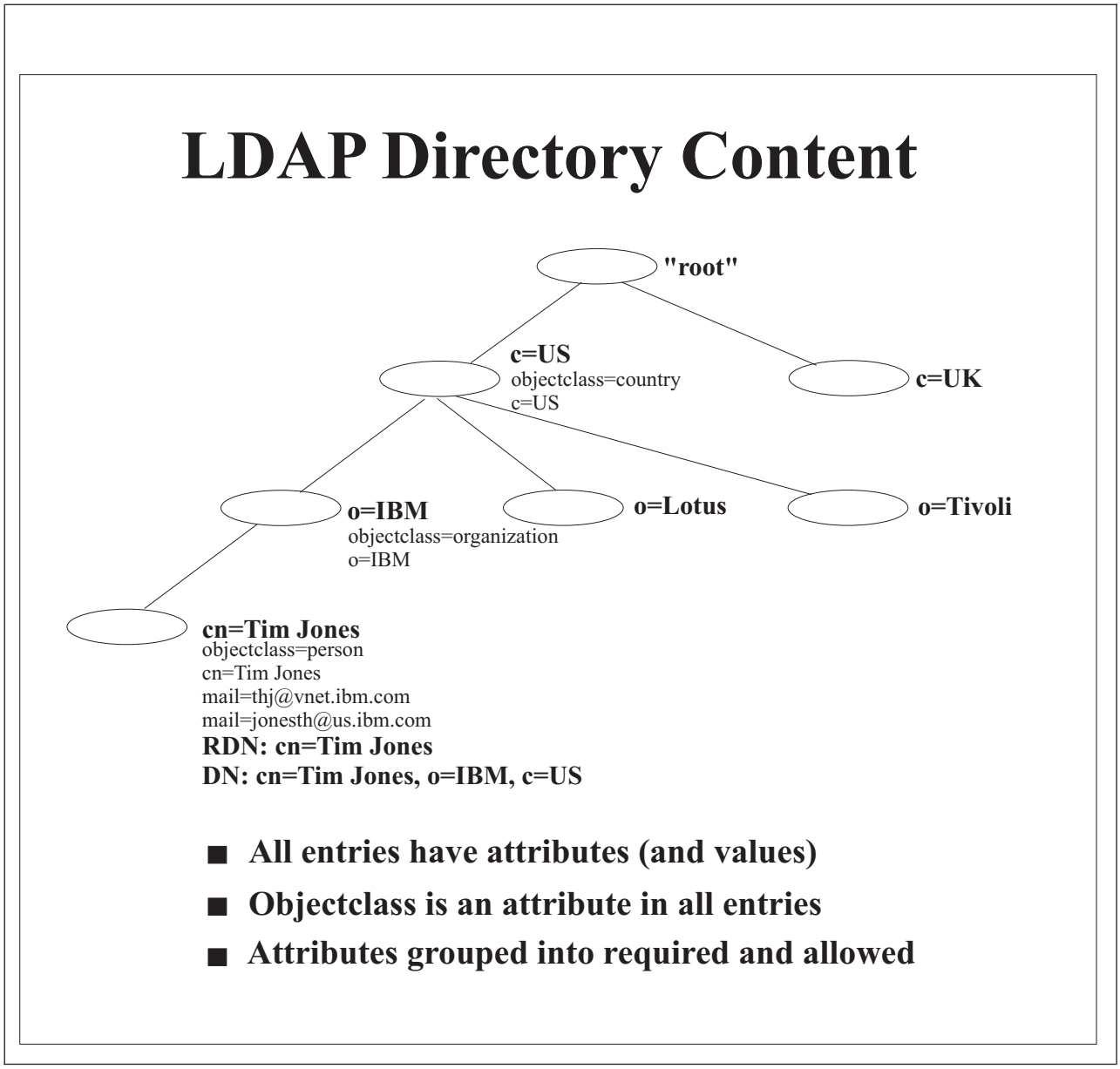


Figure 1. Directory hierarchy example

In addition, LDAP allows you to control which attributes are required and allowed in an entry through the use of a special attribute called *object class*. The values of the **objectClass** attribute determine the attributes that can be specified in the entry.

How is the information referenced?

An entry is referenced by its distinguished name, which is constructed by taking the name of the entry itself (called the *relative distinguished name*, or RDN) and concatenating the names of its ancestor entries. For example, the entry for Tim Jones in the example above has an RDN of `cn=Tim Jones` and a DN of `cn=Tim Jones, o=IBM, c=US`. The full DN format is described in IETF RFC 2253, *LDAP (V3): UTF-8 String Representation of Distinguished Names*.

The z/OS LDAP server supports different naming formats. While naming based on country, organization, and organizational unit is one method, another method is to name entries based on an organization's registered DNS domain name. Names of this form look like: `cn=Tim Smith,dc=vnet,dc=ibm,dc=com`. These naming formats can be mixed as well, for example: `cn=Tim Brown,ou=Sales,dc=ibm,dc=com`.

How is the information accessed?

LDAP defines operations for interrogating and updating the directory. Operations are provided for adding/deleting an entry to/from the directory, changing an existing entry, and changing the name of an entry. Most of the time, though, LDAP is used to search for information in the directory. The LDAP search operation allows some portion of the directory to be searched for entries that match some criteria specified by a search filter. Information can be requested from each entry that matches the criteria. The LDAP compare operation allows a value to be tested in an entry without returning that value to the client.

An example of search is, you might want to search the entire directory subtree below IBM for people with the name Tim Jones, retrieving the e-mail address of each entry found. LDAP lets you do this easily. Or you might want to search the entries directly below the `c=US` entry for organizations with the string `Acme` in their name, and that have a FAX number. LDAP lets you do this too. The section "How does LDAP work?" describes in more detail what you can do with LDAP and how it might be useful to you.

The LDAP bind operation is used to indicate to the LDAP server who is going to be making add/modify/search/compare or delete requests. The LDAP bind operation is an authentication process. This authentication process can be used by distributed applications which need to implement some form of authentication.

How is the information protected from unauthorized access?

An Access Control List (ACL) provides a means to protect information stored in an LDAP directory. ACLs are used to restrict access to different portions of the directory, specific directory entries, or information within an entry. Access control can be specified for individual users or groups.

How does LDAP work?

LDAP directory service is based on a client/server model. One or more LDAP servers contain the data making up the LDAP directory tree. An LDAP client application connects to an LDAP server using LDAP APIs and asks it a question. The server responds with the answer, or with a pointer to where the application can get more information (typically, another LDAP server). With a properly constructed namespace, no matter which LDAP server an application connects to, it sees the same view of the directory; a name presented to one LDAP server references the same entry it would at another LDAP server. This is an important feature of a global directory service, which LDAP servers can provide.

What about X.500?

LDAP was originally developed as a front end to X.500, the OSI directory service. X.500 defines the Directory Access Protocol (DAP) for clients to use when contacting directory servers. DAP has been characterized as a heavyweight protocol that runs over a full OSI stack and requires a significant amount of computing resources to run. LDAP runs directly over TCP and provides most of the functionality of DAP at a much lower cost.

An LDAP server is meant to remove much of the burden from the server side just as LDAP itself removed much of the burden from clients. If you are already running an X.500 service and you want to continue to do so, you can probably stop reading this guide, which is all about running LDAP through an LDAP server without running X.500. If you are not running X.500, want to stop running X.500, or have no immediate plans to run X.500, read on.

What are the capabilities of the z/OS LDAP server?

You can use the z/OS LDAP server to provide a directory service of your very own. Your directory can contain just about anything you want to put in it. Some of the z/OS LDAP server's more interesting features and capabilities include:

- **Multiple concurrent database instances** (referred to as backends): The LDAP server can be configured to serve multiple databases at the same time. This means that a single z/OS LDAP server can respond to requests for many logically different portions of the LDAP tree. A z/OS LDAP server can be configured to provide access to RACF, as well as store application-specific information.
- **Robust database**: The LDAP server comes with a TDBM backend database based on DB2. The TDBM database is a highly scalable database implementation.

  Note: To use TDBM, DB2 is required.
- **Loading and unloading data**: The LDAP server can load a large number of entries into a TDBM DB2 database using the **ldif2tdbm** utility. See "ldif2tdbm utility" on page 138 for more information. The LDAP server can also unload a large number of entries from a TDBM DB2 database using the **tdbm2ldif** utility. See "tdbm2ldif utility" on page 148 for more information.
- **Access control**: The LDAP server provides a rich and powerful access control facility, allowing you to control access to the information in your database or databases. You can control access to entries based on LDAP authentication information, including users and groups. Group membership can be either static, dynamic, or nested. Access control is configurable down to individual attributes within entries. Also, access controls can be set up to explicitly deny access to information. See Chapter 22, "Using access control," on page 269 on access control and Chapter 21, "Static, dynamic, and nested groups," on page 257 for more information on groups.
- **Threads**: The LDAP server is threaded for high performance. A single multi-threaded z/OS LDAP server process handles all incoming requests, reducing the amount of system overhead required.
- **Replication**: The LDAP server can be configured to maintain replica copies of its database. This master/slave replication scheme is vital in high-volume environments where a single LDAP server just does not provide the necessary availability or reliability. Peer to peer replication is also supported. See Chapter 23, "Replication," on page 289 for more information. This feature is contrasted with multiple concurrent servers.
- **Referrals**: The LDAP server provides the ability to refer clients to additional directory servers. Using referrals you can distribute processing overhead, distribute administration of data along organizational boundaries, and provide potential for widespread interconnection beyond an organization's own boundaries. See Chapter 26, "Referrals," on page 315 for more information.
- **Aliases**: An alias entry can be created in the directory to point to another entry in the directory. During search operations, an alias entry can provide a convenient public name for an entry or subtree, hiding the more complex actual name of the entry or subtree. It can also avoid the need to duplicate an entry in multiple subtrees. See Chapter 24, "Alias," on page 301 for more information.
- **Change Logging**: The LDAP server can be configured to create change log entries in the GDBM backend. Each change log entry contains information about a change to an entry in a TDBM backend or to a RACF user profile. See Chapter 25, "Change logging," on page 307 for more information.
- **Configuration**: The LDAP server configuration process can be simplified by using the **ldapcnf** configuration utility. This utility requires minimal user interaction and allows novice LDAP users to quickly configure an LDAP server. See Chapter 4, "Configuring an LDAP server using the ldapcnf utility," on page 19 for more information.

  If you do not use the **ldapcnf** utility, the LDAP server is highly configurable through a single configuration file which allows you to change just about everything you would ever want to change. Configuration options have reasonable defaults, making your job much easier. See "Creating the slapd.conf file" on page 53 for more information.
- **Secure communications**: The LDAP server can be configured to encrypt data to and from LDAP clients using the z/OS Cryptographic Services System SSL. The LDAP server supports the Start TLS extended operation to switch a non-secure connection to a secure connection. It has a variety of ciphers for encryption to choose from, all of which provide server and optionally client authentication through the use of X.509 certificates. See "Setting up for SSL/TLS" on page 46 for more information.
- **Multiple concurrent servers**: The LDAP server can be configured to permit multiple instances to serve the same DB2-based backing store at the same time. The multiple server instances may run on the same z/OS image, and they may run on multiple z/OS images in a Parallel Sysplex®. This support is

available for the TDBM and GDBM backends. This improves availability and may offer improved performance in certain configurations. See “Determining operational mode” on page 82 for more information.

- **Dynamic workload management**: The LDAP server can be configured to participate in dynamic workload management in a Parallel Sysplex by exploiting TCP/IP connection optimization. With multiple concurrent server instances configured in this way, availability is improved, as is resource utilization. In addition, performance improvements may be experienced as sysplex resource utilization is more evenly balanced across z/OS systems in the sysplex. See “Determining operational mode” on page 82 for more information.
- **Access to RACF data**: The LDAP server can be configured to provide read/write access to RACF user, group, and connection profiles using the LDAP protocol. (RACF is a component of the Security Server for z/OS.) If the RACF data is shared across the sysplex, then users, groups, and connections in the sysplex can be managed using LDAP. The LDAP server's access to RACF is managed by an additional configurable backend called SDBM. See Chapter 16, “Accessing RACF information,” on page 213 for more information.

  Note: To use SDBM for ONLY authentication (LDAP bind processing), any security manager implementing the SAF service required by the **__passwd()** function call can be used. To use SDBM for accessing and updating USER and GROUP profile information, RACF is required.
- **Retrieve Policy Director data**: The z/OS LDAP server, when using the EXOP backend, supports two LDAP extended operations, **GetDnForUserid** and **GetPrivileges**, that retrieve Policy Director data from any LDAP server. See Chapter 20, “Using extended operations to access Policy Director data,” on page 255 for more information.
- **Native authentication**: The z/OS LDAP server allows clients to bind to entries in a TDBM backend by using the system for verifying the authentication attempt. The client can perform a simple bind supplying an LDAP DN of an entry in a TDBM backend along with a security manager-maintained password. Password authentication is then performed by the security manager. See Chapter 18, “Native authentication,” on page 241 for more information.

  Note: To use native authentication, any security manager implementing the SAF service required by the **__passwd()** function call can be used.
- **LDAP Version 3 protocol support**: The LDAP server provides support for Version 3 of the LDAP protocol. This includes:
  - All protocol operations
  - Implicit bind
  - Certificate (or Simple Authentication and Security Layer) bind
  - Version 3 referrals
  - Aliases
  - Controls
  - Root DSE support
  - Internationalization (UTF-8) support
  - Modify name supported for all entries including subtree move
  - Schema publication (TDBM, SDBM, and GDBM)
  - Additional syntax support (TDBM and GDBM)
  - Online schema update capability (TDBM and GDBM)
- **Dynamic schema**: The LDAP server, when using the TDBM or GDBM backend, allows the schema to be changed dynamically through the LDAP protocol. See Chapter 14, “LDAP directory schema,” on page 165 for more information.
- **Internationalization (UTF-8) support**: The LDAP server allows storage, update and retrieval, through LDAP operations, of national language data using LDAP Version 3 protocol. See “UTF-8 support” on page 157 for more information.
- **SASL external bind and client and server authentication**: The LDAP server allows client applications to use a certificate when communicating with the server using SSL/TLS communications. In order to use

a certificate on bind, the server must be configured to perform both client and server authentication. This ensures both entities are who they claim to be. See “Setting up for SSL/TLS” on page 46 for more information.

- **SASL GSS API Kerberos bind with mutual authentication**: The LDAP server allows clients to bind to the server using Kerberos credentials. Mutual authentication is used to verify both the client and server identities. See Chapter 17, “Kerberos authentication,” on page 233 for more information.
- **SASL CRAM-MD5 and DIGEST-MD5 authentication**: The LDAP server allows clients to bind to the server using DIGEST-MD5 (RFC 2831) and CRAM-MD5 (Challenge-Response Authentication Method - RFC 2195) authentication bind methods. See Chapter 19, “CRAM-MD5 and DIGEST-MD5 Authentication,” on page 251 for more information.
- **Support for root DSE**: The LDAP server supports search operations against the Root of the Directory tree as described in IETF RFC 2251, *The Lightweight Directory Access Protocol (V3)*. The so-called Root DSE can be accessed using LDAP V3 search operations. See “Root DSE” on page 335 and *z/OS Integrated Security Services LDAP Client Programming* for more information.
- **Extended group membership searching**: The LDAP server supports extended group membership searching which allows the LDAP server to find a DN that may be a member of static and nested groups in a backend (TDBM) where the DN does not reside. The LDAP server can find the group memberships for the DNs in the other backends that are configured. See the **extendedGroupSearching** configuration file option on page 62 for more information.
- **Supported server controls**:The LDAP server supports the **manageDsaIT**, **authenticateOnly**, **IBMLDAPProxyControl**, **IBMModifyDNTimelimitControl**, **IBMModifyDNRealignDNAttributesControl**, **persistentSearch**, and **schemaReplaceByValueControl**. See Appendix D, “Supported server controls,” on page 459 for more information.
- **Supported extended operations**: The LDAP server supports the **GetDnForUserid**, **GetPrivileges**, and **changeLogAddEntryRequest** extended operations. See Appendix E, “Supported extended operations,” on page 465.
- **Password encryption**: The LDAP server allows prevention of unauthorized access to user passwords stored in the TDBM backends. See “Configuring for user password encryption” on page 50 for more information.
- **Multiple socket ports**: The LDAP server can be configured to listen for secure and nonsecure connections from clients on one or more IPv4 or IPv6 interfaces on a system. With the **listen** configuration option on the LDAP server, the hostname or the IPv4 or IPv6 address, along with the port number, can target one or multiple IPv4 or IPv6 interfaces on a system. See the **listen** configuration option on page 65 for more information.
- **Persistent search**:The LDAP server provides an event notification mechanism for applications, directories and meta directories that need to maintain a cache of directory information or to synchronize directories when changes are made to an LDAP directory. Persistent search will allow these applications to be notified when a change has occurred. See Appendix D, “Supported server controls,” on page 459 for more information.
- **Ibm-entryuuid attribute**: The LDAP server now generates a unique identifier for any entry that is created or modified and does not already have a unique identifier assigned. The unique identifier is stored in the **ibm-entryuuid** attribute. The **ibm-entryuuid** attribute is replicated to servers that support the **ibm-entryuuid** attribute. A utility is provided to create the **ibm-entryuuids** for existing entries when migrating from previous releases. See Chapter 11, “Running and using the LDAP backend utilities,” on page 135 for more information on on the LDAP entry UUID utility. See the “Configuration file options” on page 56 to configure the **serverEtherAddr** keyword in the **slapd.conf** file.
- **ibm-allMembers** and **ibm-allGroups**: The LDAP server now supports the querying of the members of static, dynamic, and nested groups in a TDBM backend via the **ibm-allMembers** operational attribute. The LDAP server also supports the querying of the static, dynamic, and nested groups that a user belongs to with the **ibm-allGroups** operational attributes.

RFCs supported by z/OS LDAP

The z/OS LDAP server supports the following IETF RFCs:

- 1738 *Uniform Resource Locators (URL)*
- 1779 *A String Representation of Distinguished Names*
- 1823 *The LDAP Application Program Interface*
- 1959 *An LDAP URL Format*
- 1960 *A String Representation of LDAP Search Filters*
- 2052 *A DNS RR for specifying the location of services (DNS SRV)*
- 2104 *HMAC: Keyed-Hashing for Message Authentication*
- 2195 *IMAP/POP AUTHorize Extension for Simple Challenge/Response*
- 2222 *Simple Authentication and Security Layer (SASL)*
- 2247 *Using Domains in LDAP/X.500 Distinguished Names*
- 2251 *Lightweight Directory Access Protocol (v3)*
- 2252 *Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions*
- 2253 *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*
- 2254 *The String Representation of LDAP Search Filters*
- 2255 *The LDAP URL Format*
- 2256 *A Summary of the X.500(96) User Schema for use with LDAPv3*
- 2279 *UTF-8, a transformation format of ISO 10646*
- 2713 *Schema for Representing Java(tm) Objects in an LDAP Directory*
- 2714 *Schema for Representing CORBA Object References in an LDAP Directory*
- 2743 *Generic Security Service Application Program Interface Version 2, Update 1*
- 2744 *Generic Security Service API Version 2: C-bindings*
- 2820 *Access Control Requirements for LDAP*
- 2829 *Authentication Methods for LDAP*
- 2830 *Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security*
- 2831 *Using Digest Authentication as a SASL Mechanism*
- 2849 *The LDAP Data Interchange Format (LDIF) - Technical Specification*
- 3377 *Lightweight Directory Access Protocol (v3): Technical Specification*

Note that although the LDAP V3 protocol RFCs are listed as supported, the specific function that z/OS LDAP supports is listed in “LDAP Version 3 protocol support” on page 8.

Chapter 2. Planning and roadmap

This chapter:

- Shows you where to find information in this book that will help you plan your directory content.
- Contains a roadmap that points you to information that may be helpful in preparing for your LDAP server configuration.

Planning directory content

Before configuring and populating your database, determine:

- What type of data you are going to store in the directory.

  You should decide on what sort of schema you need to support the type of data you want to keep in your directory. The directory server is shipped with a standard set of attribute type and object class definitions.

  Before you begin adding entries to the directory, you might need to add new attribute type and object class definitions that are customized to your data.

  Schema definition styles are specific to the backend or data store being configured. Once you have determined which backends to configure, refer to Chapter 14, “LDAP directory schema,” on page 165 or “Setting up for SDBM” on page 44 for more information.
- How you want to structure your directory data.

  Refer to Chapter 13, “Data model” for more information.
- A set of policies for access permissions.

  Refer to Chapter 22, “Using access control” for more information.

LDAP server roadmap

Table 1 is a roadmap that points you to information that may be helpful in preparing for your LDAP server configuration.

See Chapter 10, “Migrating to a z/OS LDAP server,” on page 117 if you have a previous release of the LDAP server installed on your system.

For complete instructions for installing the LDAP server product, see *z/OS Program Directory* which comes with the LDAP server tape or cartridge. Be sure to read the license agreement in *z/OS Licensed Program Specifications*, which is also included in the box.

Important

Before you proceed, review the *Memo to Users*, which describes any late changes to the procedures in this book. A printed copy is included with the LDAP server tape or cartridge.

Table 1. LDAP server roadmap

Complete?	Task	Page
If you are configuring your LDAP server for z/OS WebSphere™ Application Server (z/OS Component Broker), you must:		
	See the WebSphere documentation for details on LDAP server requirements.	
If you are configuring your LDAP server for z/OS Hardware Configuration Definition (HCD), you must:		
	See the z/OS HCD documentation for details on LDAP server requirements.	
If you are configuring your LDAP server for Resource Measurement Facility (RMF™), you must:		

Table 1. LDAP server roadmap (continued)

Complete?	Task	Page
	See the z/OS RMF documentation for details on LDAP server requirements.	
If you are configuring your LDAP server for Managed System Infrastructure (msys), you must:		
	See the z/OS msys documentation for details on LDAP server requirements.	
If you are configuring your LDAP server for z/OS Policy Director, you must:		
	See the z/OS Policy Director documentation for details on LDAP server requirements.	
If you want to see how a working LDAP server looks, you must:		
	See the **/usr/lpp/ldap/examples/sample_server** directory which contains everything necessary to set up a sample LDAP server. See the **README** file in this directory for complete instructions.	
If you are migrating from a previous release of the LDAP server, you must:		
	See the migration information.	117
If this is the first time you are installing the z/OS LDAP server, you must:		
	Read the following documents that are included in the box with the z/OS LDAP server tape or cartridge: • *z/OS Program Directory*, which contains the complete install instructions. • *z/OS Licensed Program Specifications*, which contains the license agreement. • *Memo to Users*, which describes any late changes to the procedures in this book.	
Choose a configuration method from the following options:		
	The **ldapcnf** configuration utility uses a profile file as input to generate jobs to set up the system environment and configuration. Check “Restrictions” on page 20 to decide if this method will work for your installation.	19
	If you do not use the **ldapcnf** utility, use the instructions for customizing your configuration.	53

Chapter 3. Installing and setting up related products

This chapter discusses what products are necessary to install or set up prior to configuring the z/OS LDAP server product. There are some decisions you must make depending on how you want to set up your LDAP server.

If you plan to use:	You must:	See:
TDBM or GDBM backend (based on DB2)	Install the DB2 product and set up CLI and ODBC. Note that if your LDAP server will be used **only** for accessing RACF information, it is not necessary to install DB2 or set up a DB2 database. See "Setting up for SDBM" on page 44 for information on configuring the LDAP Server for accessing RACF information.	"Installing and setting up DB2 for TDBM or GDBM" below
SDBM backend (based on RACF)	Install RACF.	"Installing RACF for SDBM and native authentication" on page 15
Program call (PC) support and the EXOP backend to support Policy Director extended operations	Install Policy Director and use SAF.	"Installing and setting up Policy Director and SAF for z/OS Policy Director support" on page 15
Protect access to your LDAP server with Secure Sockets Layer (SSL) security or Transport Layer Security (TLS)	Install z/OS Cryptographic Services System SSL.	"Installing System SSL" on page 15
Password encryption with TDBM	Install OCSF and ICSF.	"Installing OCSF and ICSF for password encryption" on page 16
Kerberos authentication	Install Kerberos.	"Installing Kerberos" on page 17
Native authentication	Install a security server.	"Installing RACF for SDBM and native authentication" on page 15

Installing and setting up DB2 for TDBM or GDBM

This section describes how to get DATABASE 2™ (DB2) running and how to run the LDAP server using the TDBM or GDBM (DB2) backend. You should also have or have access to *DB2 ODBC Guide and Reference*, *DB2 Installation Guide*, and *DB2 Application Programming and SQL Guide*.

Getting DB2 installed and set up for CLI and ODBC

Following are the steps to get DB2 installed:

1. Have your database system administrator install DB2 for OS/390 and z/OS Version 6 or later. If you will be running your LDAP server in multi-server mode on multiple images in a Parallel Sysplex, your administrator must configure a DB2 data sharing group with members on each of the z/OS images on which an LDAP server instance will run. (See "Determining operational mode" on page 82 for a description of the various operating modes in which the LDAP server may run.)

   Make sure that the SMP/E jobs are a part of the DB2 installation. See the section about installing DB2 CLI in *DB2 ODBC Guide and Reference*. Also, specify the user ID (for example, su*xxxx*) that should be granted database system administrator authority. This should be the ID you will log on with to run the SPUFI jobs to create the DB2 tables for the LDAP server. You need to find out the following information from your database administrator:
   - DB2 subsystem name. For example, DSN7.

- DB2 server location (or data source). For example, LOC1.

In order to use a local or remote DB2 database, you must include a DDF record in your Bootstrap Data Set (BSDS). That DDF record must include a LOCATION keyword and an LUNAME keyword. If you are using a DB2 database that is on the local system (including a database that is set up for DB2 data sharing) the DDF component need not be started. If you are using a DB2 database that is on a remote system, the DDF component of DB2 must be configured and started on systems using the DB2 Call Level Interface (CLI). CLI is used by the LDAP server for requesting services from DB2. (The DB2 Call Level Interface is IBM's callable SQL interface used by the DB2 family of products, based on the ISO Call Level Interface Draft International Standard specification and the Microsoft® Open Database Connectivity specification.)

It may be necessary to have your DB2 administrator set up buffer pools, TEMP space, and TEMP datasets for additional buffer pool sizes. By default, the LDAP server DB2 backing stores will use bufferpool BP0. The bufferpool choice and size (4K, 32K or other sizes) should be examined by your database system administrator to ensure they are large enough to meet the additional needs of the LDAP server, once you have loaded data into its database. The DB2 **runstats** command should be used once data is loaded so that DB2 queries are optimized.

It may also be necessary to have your DB2 administrator increase the configured DB2 limits for MAX USERS and MAX BATCH CONNECT settings to accommodate the resources required by the LDAP server. These settings are controlled by the DB2 subsystem parameters CTHREAD and IDBACK, respectively, by way of installation panel DNSTIPE. These parameters are discussed in more detail in the *DB2 Installation Guide*. The LDAP server requires the following connections to DB2:

- two connections for miscellaneous functions
- one connection for each communication thread, as defined by the **commThreads** option in the LDAP server configuration file
- one connection for each program call thread, as defined by the **pcThreads** option in the LDAP server configuration file
- one connection for each defined replica object, when replication is being used

2. Enter:

   `-dsn start db2`

   from the image console and wait for DB2 to finish the DB2 initialization. The *dsn* is the DB2 subsystem name.

   You can stop DB2 by entering:

   `-dsn stop db2`

   from the console.

   Note: This may already be done when the system is re-ipled.

3. Edit and Submit *DSNHLQ*.SDSNSAMP(DSNTIJCL) where *DSNHLQ* is the high-level qualifier used during DB2 installation. See the section on setting up DB2 CLI runtime environment in *DB2 ODBC Guide and Reference*. You must run this from the user ID that has been granted the appropriate database authorities. This step establishes the environment needed for the LDAP server to use the CLI. It is often referred to as ″binding the CLI plan″. When binding the CLI plan, it must be bound using the bind option RELEASE(COMMIT), either by default (when no RELEASE option is specified on the bind statement) or by explicitly specifying the option on the bind statement (see *DB2 Command Reference* for information on bind options and syntax). Note the plan name used when editing this job, for example DSNACLI.
4. Create (Allocate) DB2 CLI Initialization File. A sample of the CLI initialization file can be found at *DSNHLQ*.SDSNSAMP(DSNAOINI). Create your own CLI initialization file and copy *DSNHLQ*.SDSNSAMP(DSNAOINI) into it. If a dataset is used for the CLI initialization file, it must not contain sequence numbers. Refer to the section on the DB2 CLI initialization File in *DB2 ODBC Guide and Reference* for more information on the contents of this file. Figure 2 on page 15 shows a sample file. The example in Figure 2 on page 15 shows a DSNAOINI file with values based on the examples

used in this section. Items in the file that may need to be customized to your DB2 installation are in **bold** type. See your DB2 administrator for the values of these items for your installation.

```
;This is a comment line...
; Example COMMON stanza
[COMMON]
MVSDEFAULTSSID=DSN7

; Example SUBSYSTEM stanza for your DB2 subsystem name
[DSN7]
MVSATTACHTYPE=CAF
;MVSATTACHTYPE=RRSAF
PLANNAME=DSNACLI

; Example DATA SOURCE stanza for your data source
[LOC1]
AUTOCOMMIT=0
CURSORHOLD=0
CONNECTTYPE=1
```

Figure 2. Sample DSNAOINI file

Choosing the MVSATTACHTYPE

The LDAP server can be set up to use either the Call Attachment Facility (CAF) or the Recoverable Resource Manager Services Attachment Facility (RRSAF) to access DB2. See *DB2 ODBC Guide and Reference* for more information about these choices.

Installing RACF for SDBM and native authentication

In order for your LDAP server to have access to RACF data, you must have RACF installed on your system and have a license for the z/OS Security Server. RACF is part of the z/OS Security Server. Refer to the following books for information on installing and configuring RACF:

- *z/OS Program Directory*
- *z/OS Security Server RACF Security Administrator's Guide*
- *z/OS Migration*

The RACF Subsystem function of RACF must be defined and activated to allow the LDAP server to communicate with RACF through the SDBM backend. See *z/OS Security Server RACF System Programmer's Guide* for information.

Installing and setting up Policy Director and SAF for z/OS Policy Director support

In order for your LDAP server to provide z/OS Policy Director support, you must install and set up Policy Director. See *Policy Director Authorization Services for z/OS and OS/390 Customization and Use* for instructions. Policy Director support also uses the System Authorization Facility (SAF) interface which is part of the z/OS environment and is always present on your z/OS system. *z/OS Security Server RACF System Programmer's Guide* provides more information on SAF.

Installing System SSL

In order for your LDAP server to provide SSL/TLS support, you must install z/OS Cryptographic Services System SSL and use STEPLIB, LPALIB, or LINKLIST to make their libraries available. See "Setting up for SSL/TLS" on page 46 and *z/OS Crytographic Services System Secure Sockets Layer Programming* for more information regarding SSL/TLS. Also, see "Setting up for SSL/TLS" on page 46 for details on configuring and using SSL/TLS with your LDAP server.

Installing OCSF and ICSF for password encryption

The LDAP server uses the Open Cryptographic Services Facility (OCSF) to provide MD5 and SHA hashing of user passwords in the TDBM backend. The LDAP server uses both OCSF and the Integrated Cryptographic Service Facility (ICSF) to provide DES encryption and decryption of user passwords. The LDAP server does not require OCSF or ICSF to provide crypt() level encryption of user passwords. If you plan to encrypt passwords using MD5 hashing, SHA hashing, or DES encryption, or if your directory already contains passwords encrypted using one of those methods, you must install and configure the appropriate facility, or facilities, along with the LDAP server.

Note: When the LDAP server is started, it always attempts to initialize encryption with OCSF. If OCSF is not installed, this can result in a RACF message indicating insufficient access authority, such as:

```
ICH408I USER(LDAPSRV ) GROUP(OMVS    ) NAME(LDAP SERVER STARTED )
CDS.CSSM CL(FACILITY)
INSUFFICIENT ACCESS AUTHORITY
ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
```

If you do not need encryption and thus did not install OCSF, ignore this message.

OCSF

In preparation for installing OCSF for password encryption, be sure to set LIBPATH in the **/etc/ldap/slapd.envvars** file. To install and configure OCSF, refer to the configuration information in *z/OS Open Cryptographic Services Facility Application Programming*. This contains instructions on how to set up the necessary security authorizations using RACF to use OCSF. OCSF must be configured so that the user ID under which the LDAP server runs can use OCSF services. It also contains information on the Program Control necessary for OCSF. This documentation also contains instructions on how to run the installation scripts necessary to use OCSF.

Note: Although both OCSF and ICSF come with the base feature of z/OS, in the United States and Canada, an additional OCSF Security Level 3 feature must be ordered. There is no charge for this feature.

ICSF

To install, configure, and activate ICSF, your processor must have hardware cryptographic support. All new processors have hardware cryptographic support, while some older processors optionally provided this support.

Two other services of ICSF needed for DES encryption in the LDAP server are the Key Generator Utility Program (KGUP) and the Cryptographic Key Data Set (CKDS). These are needed to generate and store the key and key label needed for DES encryption of user passwords. Refer to the information about managing cryptographic keys and using the Key Generator Utility Program in *z/OS Cryptographic Services ICSF Administrator's Guide* for instructions on how to generate and store into CKDS a single-length data-encrypting key (also referred to as data key) for DES encryption and how to set up the necessary security authorizations when using RACF to protect use of the key. It is important to remember to refresh both CKDS and RACF after you make the changes. ICSF must be configured so that the user ID under which the LDAP server runs can use ICSF services.

Other parts of the ICSF book may be useful for general background information about ICSF and Cryptographic Keys.

Installing Kerberos

In order for your LDAP server to provide Kerberos support, you must install the Security Server Network Authentication and Privacy Service for z/OS which is the IBM implementation of Kerberos Version 5. See *z/OS Integrated Security Services Network Authentication Service Administration* for more information regarding Kerberos.

A sample Kerberos configuration file is provided in **/etc/skrb**. Refer to *z/OS Integrated Security Services Network Authentication Service Administration* for details on setting up this file.

Chapter 4. Configuring an LDAP server using the ldapcnf utility

The LDAP configuration utility, **ldapcnf**, simplifies and automates the LDAP server configuration process for GDBM, TDBM, or SDBM backends. The following table shows where to find specific information about the LDAP configuration utility in this chapter.

Overview of the LDAP configuration utility

The LDAP configuration utility helps you configure new LDAP server instances with minimal user interaction.

The LDAP configuration utility takes a profile file as input and generates a set of output members in a data set to facilitate an LDAP server configuration. The profile file is targeted for the System Administrator (or System Programmer) and the LDAP Administrator and it contains statements that must be updated with appropriate values. The LDAP configuration utility generates a series of JCL members, configuration files, and a procedure to start the LDAP server. The JCL jobs are segregated based on typical administrative roles in a z/OS installation and contain the required commands to configure the z/OS components used by the LDAP server. Each administrator is responsible for reviewing and submitting their JCL job. After all JCL jobs are submitted, each administrator is responsible for reviewing their job's output and addressing any errors that may have occurred. Once all JCL jobs have completed successfully, the LDAP server can be started.

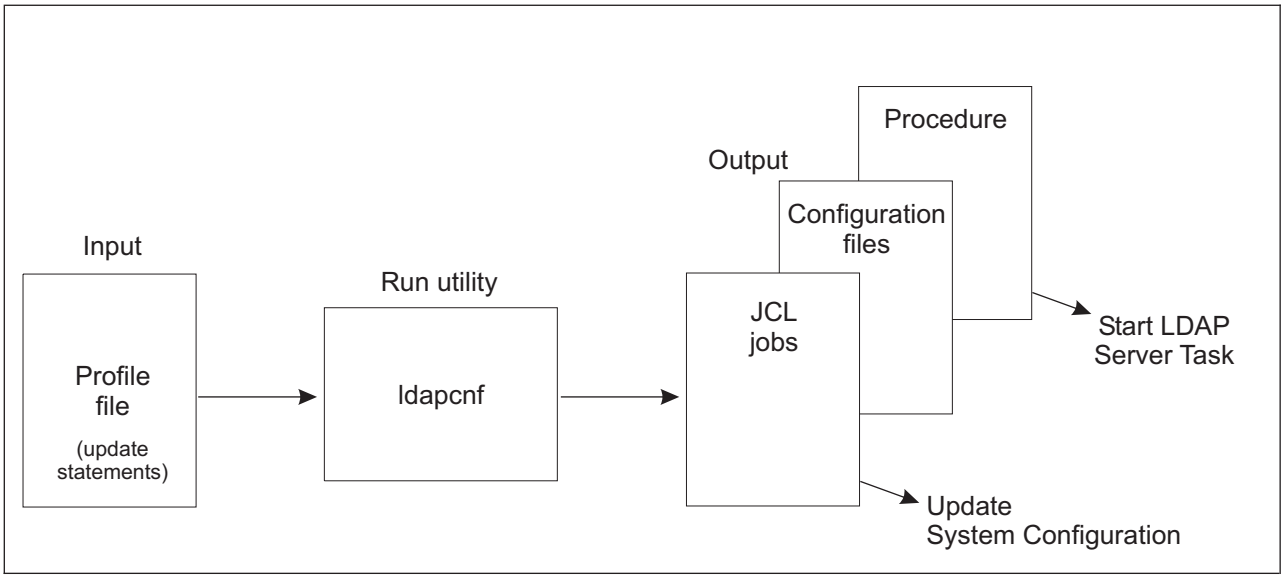


Figure 3. LDAP configuration utility overview

The minimal user interaction with the utility and the jobs it produces to update the required z/OS components results in a simplified approach to LDAP configuration. This approach allows novice LDAP users and administrators and even novice z/OS users to quickly deploy an LDAP server. In addition, the utility does not restrict the configuration of advanced LDAP features, such as referrals, replication, password encryption, and sysplex setup. See “Specifying advanced configuration options with the ldapcnf utility” on page 29 for more information.

Capabilities

Following are the capabilities of the LDAP configuration utility:

- Allows for the configuration of a TDBM (DB2-based) backend, an SDBM (RACF-based) backend, an Extended operations (EXOP) backend, and a change log GDBM (DB2-based) backend.
- Generates JCL jobs to accomplish the updates of all the z/OS components required for an LDAP server.
- Can configure advanced LDAP server features, including:
  - Password encryption (**ldapcnf** does not generate certificates or passwords)
  - Referrals
  - Replication
  - Change logging
  - Secure Sockets Layer (SSL) or Transport Layer Security (TLS) (**ldapcnf** does not generate certificates or passwords)
  - Kerberos authentication
  - Native authentication
  - Extended operations (EXOP) backend (used for accessing Policy Directory information)

Restrictions

Following are some restrictions regarding the LDAP configuration utility:

- Generates a procedure; therefore, the LDAP server must run as a started task.
- Assumes that RACF is the security server in use. However, if RACF is not the security server in use, **ldapcnf** could still be used. The resulting RACF JCL job will need to be converted to properly update the security server in use.
- Does not handle multiple TDBM (DB2-based) backends.
- All values in the input files must be less than 66 bytes in length and must contain only printable characters in the IBM-1047 code page.

- Cannot extend or enhance an existing LDAP server configuration. Furthermore, any manual updates to the output that the utility produces will be lost if you run the utility again with the same output data set.
- Does not support configuration for an LDAP server to listen on more than one secure port.
- Does not support configuration for an LDAP server to listen on more than one non-secure port.

If you cannot use the **ldapcnf** utility because of one or more of these restrictions, see Chapter 5, “Configuring an LDAP server without the ldapcnf utility,” on page 33 for information on alternate methods you can use to configure your LDAP server.

Using the ldapcnf utility

The **ldapcnf** utility is used to generate jobs to set up the system environment and configuration for a new LDAP server. This utility is installed into the **/usr/lpp/ldap/sbin** directory.

Format

ldapcnf -i *profile_file*

where:

profile_file
Specifies the input file that contains statements necessary to configure the LDAP server. See "Input file description" for more details about this file.

Example

Following is an example using **ldapcnf**, where **ldap.profile** is in the **/home/u** directory.

```
ldapcnf -i /home/u/ldap.profile
```

Input file description

The input file, **ldap.profile**, shipped in the **/usr/lpp/ldap/etc** directory, contains the settings necessary to set up an LDAP server. You must copy the **ldap.profile** file and then modify it before the LDAP configuration utility, **ldapcnf**, can be run.

In this file there are statements containing a keyword and value which must have the appropriate value for the target system being configured. Figure 4 shows a sample portion of the **ldap.profile** file:

```
# LDAPUSRID <user_id>
#
# Description:
#    User ID for the LDAP server to run under.
#
# Note:
#    This variable's value must be capitalized.
# ---------------------------------------
LDAPUSRID='GLDSRV'
```

Figure 4. Sample portion of ldap.profile

The LDAPUSRID statement, as shown above, has a pre-assigned value of GLDSRV. Above the statement there is some commentary describing the statement and its usage.

Most of the statements in the **ldap.profile** are required and those that are not required are labelled as optional. Some statements in the **ldap.profile** have pre-assigned values; however, they may not be valid on the target system being configured. Values must be provided for all required statements in the **ldap.profile** file.

The **ldap.profile** file embeds three other advanced input files. Information about these files can be found in "Specifying advanced configuration options with the ldapcnf utility" on page 29. All of the input files are in the same format as an environment variable file.

Usage

1. The output from **ldapcnf** is written to an output data set that you specify in **ldap.profile**. If the data set does not exist, the utility allocates the output data set for you.

2. The utility allows the configuration of an LDAP server which uses SSL/TLS. (See “Setting up for SSL/TLS” on page 46 for details.) It does not, however, automate the process of generating SSL/TLS certificates.
3. The utility allows the configuration of an LDAP server which uses password encryption. It does not, however, automate the process of generating encryption keys. (See “Configuring for user password encryption” on page 50 for more information.)
4. Verify that the SYS1.SIEALNKE data set containing the LDAP code is in the LINKLIST. If it is not in LINKLIST, then STEPLIB must be used to locate this data set.

   The **ldapcnf** utility does not provide an interface for adding STEPLIB statements to the started task procedure that it generates. Therefore, if an administrator wishes to add STEPLIB statements to the started task procedure, the started task procedure must be manually updated and the following must occur:

   - The data sets specified in the new STEPLIB statements must be APF authorized.
   - When submitting the PRGMCTRL JCL job, the data sets specified in the new STEPLIB statements must be in the program control data set list.
   - The user ID specified on the LDAPUSRID statement in the **ldap.profile** file must have read access to the data sets specified in the new STEPLIB statements.
5. The APF JCL job will not work on a system using JES3. JES3 users need to manually enter the following operator command in place of submitting the APF JCL job:

   - In SDSF, enter:

     `/SET PROG=PROG`*`suffix`*
   - From the operator’s console, enter:

     `SET PROG=PROG`*`suffix`*

   The *suffix* above is specified on the PROG_SUFFIX statement in the **ldap.profile** file.
6. The PRGMCTRL and RACF jobs that **ldapcnf** generates require that the definitions listed below exist in RACF prior to submission. If the definitions do not exist, the jobs will contain RACF errors in their output.
   a. To ensure that all required data sets are program controlled, the PRGMCTRL job requires that the PROGRAM ** definition exists in RACF.
   b. To ensure that the user ID that appears on the LDAPUSRID statement in the **ldap.profile** file has read permission on all required data sets, the RACF job requires that data set definitions exist for the following data sets in RACF.
      - *CEEHLQ*.** (where *CEEHLQ* appears on the CEEHLQ statement in the **ldap.profile** file)
      - *GLDHLQ*.** (where *GLDHLQ* appears on the GLDHLQ statement in the **ldap.profile** file)
      - *GSKHLQ*.** (where *GSKHLQ* appears on the GSKHLQ statement in the **ldap.profile** file)
      - *DSNHLQ*.** (where *DSNHLQ* appears on the DSNHLQ statement in the **ldap.profile** file)
      - *CBCHLQ*.** (where *CBCHLQ* appears on the CBCHLQ statement in the **ldap.profile** file)
      - *OUTPUT_DATASET_HLQ*.** (where *OUTPUT_DATASET_HLQ* is the first level high-level qualifier of the data set name that appears on the OUTPUT_DATASET statement in the **ldap.profile** file)

        Note that **ldapcnf** does not parse multi-level high-level qualifiers for the primary high-level qualifier. Thus, **ldapcnf** requires that data set definitions in RACF use the full high-level qualifiers as specified on the high-level qualifier statements in the **ldap.profile** file.

        Also note that the server will operate properly even if the definitions required by the RACF JCL job do not exist, given that the user ID that appears on the LDAPUSRID statement in the **ldap.profile** file has read permission on all required data sets.
7. Administrators with the appropriate authorizations must submit the JCL jobs generated by **ldapcnf** on the target system.
8. When running **ldapcnf** from an **rlogin** session, if OUTPUT_DATASET has not been pre-allocated, the script will try to free it after allocation. The free may exit with a return code 12, indicating that it is not

currently allocated. The **rlogin** environment implicitly frees the data set when it is allocated, which is the cause of this error. This error can be ignored if running under this environment.

9. If an error occurs when submitting an **ldapcnf** output JCL job or, if prior to submission, an administrator considers a value within the JCL job unsatisfactory, the administrator should not modify the JCL job directly. Instead, the administrator should update the appropriate profile files and perform all of the steps outlined in “Steps for configuring an LDAP server” on page 26 again.

   To help determine the statements within a profile file that the administrator may need to update, at the top of every file generated by **ldapcnf** there is a listing of statements that **ldapcnf** used to create the output file. The administrator can use this listing to determine the exact statement within a profile file that should be updated. Note that when resubmitting all the JCL jobs **ldapcnf** creates, many times JCL jobs for other components may have errors due to the duplication of a previous update. These messages may be ignored.
10. If a statement's value requires a length greater than 65 within the generated SLAPDCNF, the LDAP Administrator can move the SLAPDCNF member out of the output data set into a data set where the record length is greater than 80 bytes and update the member in the new data set. Then, the System Administrator must update the CONFIG DD card in the generated procedure to point to the new data set.
11. If the utility is configuring an SDBM backend and password encryption, the PRGMCTRL JCL job will use a shell script in **/tmp** called **gldOcsfApf.sh.** *pid* (where *pid* is the process ID of which the **ldapcnf** shell was run under) which is generated and placed in **/tmp** by the **ldapcnf** utility. This shell script must be available on the target system where the PRGMCTRL JCL job is submitted. This file should not be deleted until the PRGMCTRL member is submitted.

    Also note that under the same conditions, the generated PRGMCTRL JCL job causes two HFS output files to be created: **/tmp/gldCmd1.out.** *pid* and **/tmp/gldCmd1.err**. These files report information about the success of the **gldOcsfApf.sh** shell.

    If there were no problems encountered in the shell script when the PRGMCTRL JCL job is submitted, the shell script and the two output files will be deleted.
12. The profile files do not replace the LDAP server configuration file; they are used to create an LDAP server configuration file to run the LDAP server.
13. Error messages result if required statements are not assigned values in the input files or if simplified syntax checking fails. An error message also results if the same database name or database owner is specified for both GDBM and TDBM backends. GDBM and TDBM cannot share a database.
14. It is recommended that you make all updates through the input files, running the utility again to recreate the jobs. Otherwise, if the generated JCL jobs are manually updated, those updates will be lost if the utility is run again using the same output data set.
15. Be sure to use a different output data set than is currently being used by other LDAP servers.
16. The DBCLI, DSNAOINI, TDBSPUFI, and GDBSPUFI output members are only created when a TDBM backend or the GDBM backend is being configured.

Configuration roles and responsibilities

The output from the LDAP configuration utility consists of jobs and configuration files that finalize the LDAP server configuration. These jobs segregate z/OS updates based on typical administrative roles, allowing each administrator to control their component's updates. The typical administrative roles that are assumed to exist to configure an LDAP server are:

- System Administrator (or System Programmer)
- Database Administrator
- LDAP Administrator
- Security Administrator

Each administrator is responsible for updating input files in addition to reviewing and submitting jobs in the output members that the LDAP configuration utility produces for their component, as shown in Table 2 on page 25.

Table 2. LDAP configuration utility roles and responsibilities

Role	Responsibility	Input file name/type	Output members
System Administrator (or System Programmer)	APF authorization	**ldap.profile** (main)	• APF • PROG*suffix* (*suffix* is specified on the PROG_SUFFIX statement in **ldap.profile**)
Database Administrator	DB2, CLI	**ldap.db2.profile** (advanced)	• TDBSPUFI • GDBSPUFI • DBCLI • DSNAOINI
LDAP Administrator	LDAP server, Kerberos authentication, native authentication	**ldap.slapd.profile** (advanced)	• *user_id* procedure (user ID is specified on the LDAPUSRID statement in **ldap.profile**) • SLAPDENV • SLAPDCNF
Security Administrator	RACF SSL/TLS, password encryption	**ldap.racf.profile** **ldap.slapd.profile** (both files are advanced)	• RACF • PRGMCTRL
Note about Security Administrator: If configuring SDBM and password encryption, the Security Administrator must have read/write authority on all files in the **/usr/lpp/ocsf/lib** and **/usr/lpp/ocsf/addins** directories.			

Figure 5 on page 26 is a graphical representation showing the administrative roles, input files, and output members for **ldapcnf**.

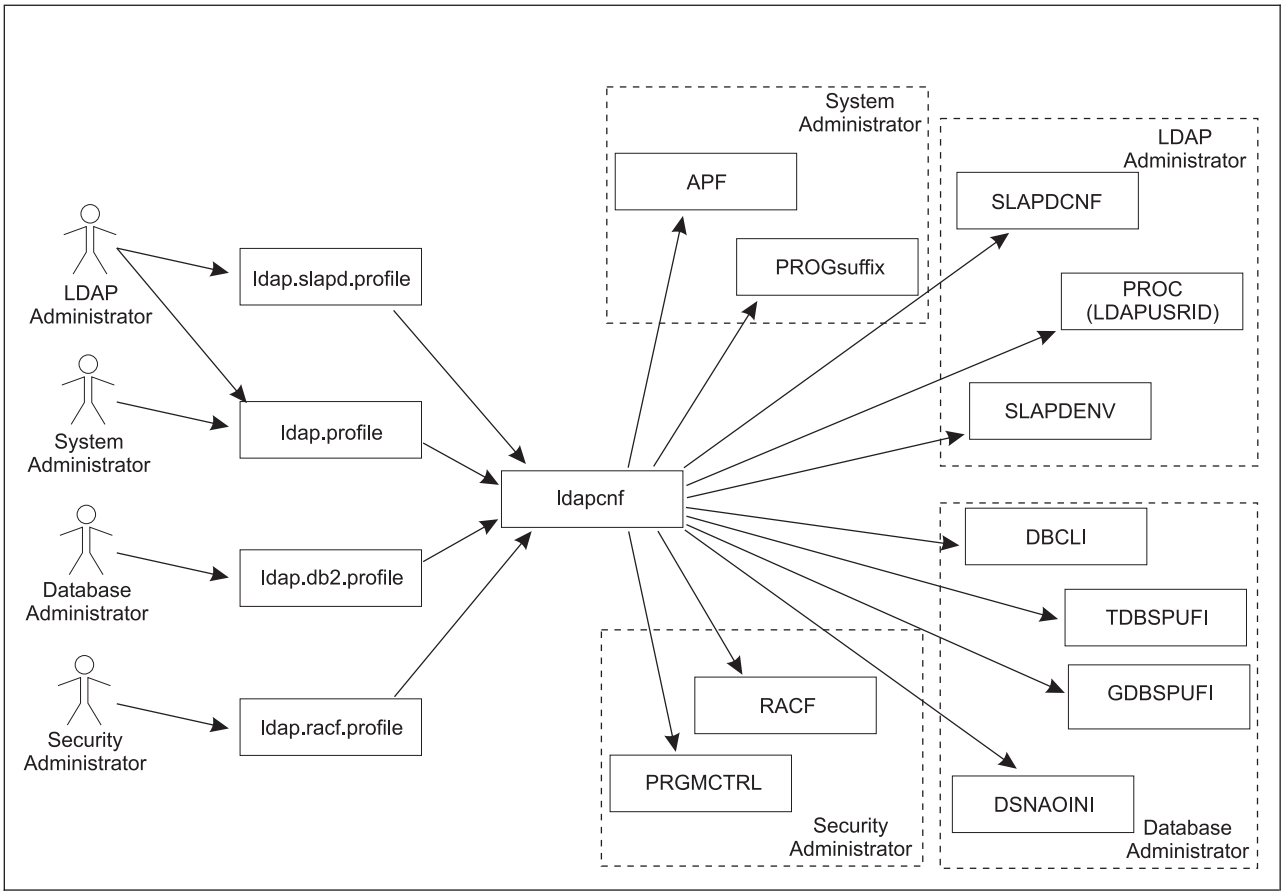


Figure 5. LDAP configuration utility roles and responsibilities

Steps for configuring an LDAP server

Use the following steps to configure with the configuration utility:

1. Copy the **ldap.profile** file, found in **/usr/lpp/ldap/etc**, to a local directory and update it according to the commentary found in the file. (If you need to update the advanced configuration files mentioned in Table 2 on page 25, you need to copy those files as well. See "Specifying advanced configuration options with the ldapcnf utility" on page 29 for more details on these files.)

   Some statements in **ldap.profile** do not have any pre-assigned values but are required for successful configuration. These are noted in the file. Assign values to all of these required statements, referring to information in the file above each statement for assistance. The intended audience of the **ldap.profile** file is the System Administrator (or System Programmer) and the LDAP Administrator. The file contains information required from both administrators.

   Note: Some statement values are case-sensitive and are denoted accordingly. Be sure to set up the editor to allow both upper and lower case letters to be specified.

2. Run the **ldapcnf** utility to generate members. (See "Using the ldapcnf utility" on page 22 for detailed information.) The generated members will be placed in the output data set specified on the OUTPUT_DATASET statement in the **ldap.profile** file.

   The utility creates:
   - JCL jobs for each role
   - SLAPDCNF member which is the LDAP server configuration file
   - SLAPDENV member which is the LDAP server environment variable file
   - PROG member needed for APF authorization

- Procedure needed to start the LDAP server
- DSNAOINI configuration file for DB2 CLI
- TDBSPUFI DB2 SQL statements for TDBM
- GDBSPUFI DB2 SQL statements for GDBM

3. Copy members and submit jobs.
   a. Copy the LDAP server started task procedure from the output data set to the target system's procedure library. The name of the LDAP server started task procedure will be the name of the LDAP user ID specified on the LDAPUSRID statement in the **ldap.profile** file. The pre-assigned name of the LDAP user ID is **GLDSRV**.
   b. Copy the generated PROG*suffix* member (where *suffix* is specified on the PROG_SUFFIX statement in the **ldap.profile** file) from the output data set to the target system's PARMLIB.
   c. Submit the following generated JCL jobs that can be found in the output data set in the following order:
      1) RACF member
      2) APF member
      3) DBCLI member, if TDBM or GDBM is being configured

         Note: Be sure DB2 is started before submitting this job.
      4) PRGMCTRL member

         The PRGMCTRL member is only required if one of the following scenarios exist:
         - Password encryption is being configured **or**
         - An SDBM backend is being configured **and**
           - Program Control is active
4. Through the DB2 SPUFI interactive tool, submit the TDBSPUFI member, if a TDBM backend is being configured, and the GDBSPUFI member, if the GDBM backend is being configured.
5. Start the LDAP server. The LDAP server can be started from SDSF or from the operator's console.

   Note: The name of the LDAP server procedure is the same as the user ID specified on the LDAPUSRID statement. The pre-assigned value is **GLDSRV**.

   To start the LDAP server in SDSF, enter:

   ```
   /s user_id
   ```

   To start the LDAP server from the operator's console, enter:

   ```
   s user_id
   ```
6. Finalize set-up of the LDAP server.
   a. Modify the schema entry for the TDBM backend, if configured.

      Copy the **schema.user.ldif** file, found in the **/usr/lpp/ldap/etc** directory, into a local directory. Fill in the TDBM suffix in the line (shown below) which is near the top of the **schema.user.ldif** file.

      ```
      dn: cn=schema, <suffix>
      ```

      The suffix is the same value used in the TDBM_SUFFIX statement in the **ldap.profile** file. Here is an example with the suffix filled in:

      ```
      dn: cn=schema, o=Your Company
      ```

      Use the following **ldapmodify** utility (z/OS shell version) and the parameters shown to modify the schema entry.

      ```
      ldapmodify -h ldaphost -p ldapport -D binddn -w passwd -f file
      ```

      where:

      ldaphost
      Is the host name of the system the LDAP server is running.

ldapport
Is the TCP/IP port on which the LDAP server is running. The port is specified in an advanced profile file, **ldap.slapd.profile**, on the PORT statement. The pre-assigned value is 3389.

binddn Is the administrator DN of the LDAP server. The administrator DN was specified in the **ldap.profile** file on the ADMINDN statement. The pre-assigned value is `"cn=LDAP Administrator"`.

passwd
Is the administrator password of the LDAP server. The administrator password was specified in the **ldap.profile** file on the ADMINPW statement. The pre-assigned value is `"secret"`.

file Is a file containing modifications to the schema entry in LDIF format. More information about the schema can be found in Chapter 14, “LDAP directory schema,” on page 165.

Following is an example of using **ldapmodify** to modify the schema entry:

```
ldapmodify -h myhost -p 3389 -D "cn=LDAP Administrator" -w secret -f /home/u/schema.user.ldif
```

This example assumes the schema file was copied to the **/home/u** directory and updated.

More information about **ldapmodify** can be found in *z/OS Integrated Security Services LDAP Client Programming*.

Multiple schemas may need to be loaded before applications that use the directory will work. For example, in addition to the **schema.user.ldif** schema file, it is common for directory applications to require the elements defined in the **schema.IBM.ldif** schema file.

b. Load the suffix entry for the TDBM backend, if configured. (The suffix entry is specified in the **ldap.profile** file on the TDBM_SUFFIX statement.)

Notes:

1) This step can be ignored if, once the LDAP server has been started, the LDAP Administrator intends to load data into the directory from an LDAP Data Interchange Format (LDIF) file that contains the suffix entry.
2) If you intend to load large amounts of data in LDIF format into the directory, see “ldif2tdbm utility” on page 138 for instructions on using the **ldif2tdbm** utility. In this case, do not load the suffix entry separately. Include the suffix instead with the rest of the entries to be loaded by **ldif2tdbm**.

Use the following **ldapadd** utility (z/OS shell version) and the parameters shown to load the suffix entry.

ldapadd -h *ldaphost* **-p** *ldapport* **-D** *binddn* **-w** *passwd* **-f** *file*

where:

ldaphost
Is the host name of the system the LDAP server is running.

ldapport
Is the TCP/IP port on which the LDAP server is running. The port is specified in an advanced profile file, **ldap.slapd.profile**, on the PORT statement. The pre-assigned value is 3389.

binddn Is the administrator DN of the LDAP server. The administrator DN is specified in the **ldap.profile** file on the ADMINDN statement. The pre-assigned value is `"cn=LDAP Administrator"`.

passwd
Is the administrator password of the LDAP server. The administrator password was specified in the **ldap.profile** file on the ADMINPW statement. The pre-assigned value is `"secret"`.

file Is a file containing the suffix entry in LDIF format. The distinguished name of the suffix

entry must equal the value that appears on the TDBM_SUFFIX statement in the **ldap.profile** file. More information about LDIF format can be found in “Using LDIF format to represent LDAP entries” on page 328.

Following is an example of using **ldapadd** to load the suffix entry:

```
ldapadd -h myhost -p 3389 -D "cn=LDAP Administrator" -w secret -f suffix.ldif
```

More information about **ldapadd** can be found in *z/OS Integrated Security Services LDAP Client Programming*.

c. Set an appropriate ACL for controlling access to change log entries for the GDBM backend, if configured. See Chapter 25, “Change logging,” on page 307 for more information.

To confirm the LDAP server is configured and ready for client requests, see “Configuration confirmation.”

To load the data in LDIF format into a TDBM directory, you can use **ldif2tdbm** or **ldapadd**. However, if you intend to load more than 100,000 directory entries, use **ldif2tdbm**.

Configuration confirmation

Following is an optional Installation Verification Procedure (IVP) to confirm that the LDAP server configuration was successful.

Run the **ldapsearch** utility (z/OS shell version) with the following parameters to verify the configuration.

ldapsearch -h *ldaphost* -p *ldapport* **-V 3 -s base -b "" "objectclass=*"**

where:

ldaphost
: Is the host name of the system the LDAP server is running.

ldapport
: Is the TCP/IP port on which the LDAP server is running. The port is specified in an advanced profile file, **ldap.slapd.profile**, on the PORT statement. The pre-assigned value is 3389.

The **-V 3** specifies LDAP Version 3 protocol, the **-s base** specifies the base scope for the search, and the **-b** *""* specifies the root DSE as the base.

The result of this search is a list of all naming contexts supported by the LDAP server. For example, if both TDBM and SDBM are configured, the result of the search is both naming contexts (suffixes) listed.

Following is an example using **ldapsearch** to verify a configuration:

```
ldapsearch -h myhost -p 3389 -V 3 -s base -b "" "objectclass=*"
```

If the naming context is not returned, an error message is returned indicating a problem.

More information about **ldapsearch** can be found in *z/OS Integrated Security Services LDAP Client Programming*.

Specifying advanced configuration options with the ldapcnf utility

There are advanced configuration options specified in the following input files:

- **ldap.db2.profile** (DB2 input file)
- **ldap.racf.profile** (RACF input file)
- **ldap.slapd.profile** (SLAPD input file)

These advanced profile files are located in the **/usr/lpp/ldap/etc** directory and are all included by the **ldap.profile** file. Every statement contains a pre-assigned value in the advanced profile files.

To modify these optional statements:

1. Copy the desired files to a local directory and update them. Each input file should be modified by the appropriate administrator (see Table 2 on page 25).
2. Update the **ldap.profile** to correctly include those modifications. Near the end of **ldap.profile** there are three statements:

   ```
   ${SOURCE_CMD} ${USR_LPP_ROOT}/usr/lpp/ldap/etc/ldap.slapd.profile
   ${SOURCE_CMD} ${USR_LPP_ROOT}/usr/lpp/ldap/etc/ldap.db2.profile
   ${SOURCE_CMD} ${USR_LPP_ROOT}/usr/lpp/ldap/etc/ldap.racf.profile
   ```

   Update these statements to show the new paths of the files you modified. Here is an example where the modified versions of **ldap.slapd.profile** and **ldap.db2.profile** are in different directories (**/home/u**), and **ldap.racf.profile** was not changed:

   ```
   ${SOURCE_CMD} /home/u/ldap.slapd.profile
   ${SOURCE_CMD} /home/u/ldap.db2.profile
   ${SOURCE_CMD} ${USR_LPP_ROOT}/usr/lpp/ldap/etc/ldap.racf.profile
   ```

Advanced configuration options may require additional instructions not covered by the LDAP configuration utility. The following table provides references for those instructions.

Configuration option	More information
Referrals	Chapter 26, “Referrals,” on page 315
Replication	Chapter 23, “Replication,” on page 289
Password encryption	“Configuring for user password encryption” on page 50
Multi-server	“Determining operational mode” on page 82
Kerberos authentication	Chapter 17, “Kerberos authentication,” on page 233
Native authentication	Chapter 18, “Native authentication,” on page 241
CRAM-MD5 and DIGEST-MD5 authentication	Chapter 19, “CRAM-MD5 and DIGEST-MD5 Authentication,” on page 251
Extended operations to access Policy Director data	“Setting up for extended operations” on page 46
Entry UUID support	**serverEtherAddress** option on page 73
Change logging	Chapter 25, “Change logging,” on page 307
Other LDAP server options	Chapter 8, “Customizing the LDAP server configuration,” on page 53

Notes:

1. If the UID in the **ldap.racf.profile** (specified on the on LDAPUID statement) is greater than 0 and the *port* or the *secureport* in the **ldap.slapd.profile** file (specified on the PORT and SECUREPORT statements, respectively) is less than 1024, the PORT statements generated in the SLAPDCNF member commentary must be added to the **profile.tcpip** file on the target system. These PORT statements are located in the commentary directly above the *port* and *secureport* variables in the generated SLAPDCNF member. See *z/OS Communications Server: IP Configuration Guide* for more information on the **profile.tcpip** file.
2. When using **ldapcnf** to configure an LDAP server with Kerberos enabled, the user ID that the server runs under is created with a temporary password. This temporary password is then immediately removed. This is required to complete the configuration of an LDAP server with Kerberos enabled. See Chapter 17, “Kerberos authentication,” on page 233 for more details.

Setting the time zone

The LDAP server uses time values returned by the operating system when it records server activity or when it generates LDAP trace records. The LDAP server assumes that time values are in Universal Time Coordinated (UTC) format. The UTC time value is mapped to a (local) time zone value as specified by the TZ environment variable. By default, TZ is set to GMT0.

To change the time zone value, you need to edit the **ldap.slapd.profile** file. See “Specifying advanced configuration options with the ldapcnf utility” on page 29 for more information about updating **ldap.slapd.profile**. In **ldap.slapd.profile**, uncomment the TIMEZONE variable and set the value as desired. For more information about time zones, see *z/OS C/C++ Programming Guide*, Chapter 7.4.

LDAP server configuration for other z/OS components or products

The LDAP Configuration Utility can be used to configure an LDAP server for use by other z/OS components or products, such as Managed System Infrastructure (msys). Some assigned values in the input file, or the advanced input files, may not be valid for an LDAP server required for other z/OS components or products.

For msys, there is a set of input files available that can be used to configure an LDAP server which will be used by msys. The input files can be found in **/usr/lpp/ldap/etc** and have the following names:

- **ldap.msys.profile**
- **ldap.msys.db2.profile**
- **ldap.msys.racf.profile**
- **ldap.msys.slapd.profile**

The **ldap.msys.profile** file corresponds to the **ldap.profile** file described in the previous sections in this chapter. Substitute **ldap.msys.profile** for **ldap.profile** in the instructions in “Using the ldapcnf utility” on page 22 and “Steps for configuring an LDAP server” on page 26 to modify **ldap.msys.profile**.

In addition to loading the **schema.user.ldif** file (explained in step 6 on page 27), you must also load the **schema.IBM.ldif** for msys.

Chapter 5. Configuring an LDAP server without the ldapcnf utility

This chapter lists the necessary steps involved in configuring your LDAP server if you do not use the **ldapcnf** utility. It may be necessary for you to use this method instead of the **ldapcnf** utility, as all LDAP configuration scenarios cannot be set up with the **ldapcnf** utility. More information about the **ldapcnf** utility is in Chapter 4, "Configuring an LDAP server using the ldapcnf utility," on page 19.

This chapter contains:

- A roadmap which provides LDAP server configuration steps based on which backends and options you choose to configure, such as:
  - SDBM backend (RACF-based)
  - TDBM backends (general purpose directory, DB2-based)
  - GDBM backend (change log directory, DB2-based)
  - EXOP backend for accessing Policy Director data
  - Secure Sockets Layer (SSL) or Transport Layer Security (TLS)
  - Password encryption
  - Kerberos authentication
  - Native authentication
- A list of configuration variables and their interactions
- Setting the time zone

LDAP server configuration roadmap

Table 3 lists the set up and configuration tasks you must complete depending on which backends and options your LDAP server needs.

Table 3. LDAP server configuration roadmap

Complete?	Task	Page
If you are configuring an SDBM (RACF-based) backend, you must:		
	Install RACF	15
	Set up the User ID and Security for the LDAP server	37
	Set up the LDAP server for SDBM	44
	Configure the LDAP server	53
	Run the LDAP server	101
	See other SDBM-specific information	213
If you are configuring a TDBM (DB2-based) backend, you must:		
	Install and set up DB2	13
	Set up the User ID and Security for the LDAP server	37
	Create the DB2 database and table spaces for TDBM	42
	Set up the schema for TDBM	165
	Configure the LDAP server	53
	Load the data into the LDAP server for TDBM	138
	Run the LDAP server	101
If you are configuring a GDBM (DB2-based) backend, you must:		
	Install and set up DB2	13
	Set up the User ID and Security for the LDAP server	37

Table 3. LDAP server configuration roadmap (continued)

Complete?	Task	Page
	Create the DB2 database and table spaces for GDBM	42
	Configure the LDAP server	53
	Run the LDAP server	101
If you are configuring an EXOP (extended operation) backend, you must:		
	Install Policy Director	15
	Set up the User ID and Security for the LDAP server	37
	Set up the LDAP server for EXOP	46
	Configure the LDAP server	53
	Run the LDAP server	101
If your LDAP server is going to support Secure Sockets Layer (SSL) or Transport Layer Security (TLS), you must:		
	Install and set up System SSL	15
	Locate System SSL and protect the environment for use of SSL/TLS	39
	Set up the LDAP server for SSL/TLS	46
	Configure the LDAP server	53
	Run the LDAP server	101
If your LDAP server is going to use password encryption, you must:		
	Install OCSF and ICSF	16
	Set up the LDAP server for password encryption	50
	Configure the LDAP server	53
	Run the LDAP server	101
If your LDAP server is going to support Kerberos Authentication, you must:		
	Install and configure Kerberos	17
	Start the KDC	233
	Update the Kerberos segment of the LDAP server's user ID	233
	Generate the LDAP server's key table file (optional)	233
	Configure the LDAP server	53
	Run the LDAP server	101
If your LDAP server is going to support native authentication, you must:		
	Install RACF or other security server	15
	Configure the LDAP server	53
	Run the LDAP server	101

Preparing for configuration variable interactions

Some of the variables involved in configuring the LDAP server and its related products are used in more than one file or configuration step. It is essential that the same value be used each time the variable is referenced. The following table lists the interactions of each such variable, for each backend.

Table 4. Configuration variable interactions

Variable	Used in:
TDBM or GDBM Backend	

Table 4. Configuration variable interactions (continued)

Variable	Used in:
DB2 subsystem ID	• SYSTEM(DSN) value in CLI bind JCL, DSNTIJCL (see Step 3 on page 14) • MVSDEFAULTSSID and SUBSYSTEM values in CLI initialization file, DSNAOINI (see Step 4 on page 14)
Plan name	• PLAN value in CLI bind JCL, DSNTIJCL (see Step 3 on page 14) • PLANNAME value in CLI initialization file, DSNAOINI (see Step 4 on page 14) • The *zzz* value in SQL commands to grant permissions to LDAP user (see Step 5 on page 43)
Database name	• -DDDDDDDD- value in SPUFI script to create database, *GLDHLQ*.SGLDSAMP(TDBMDB) (see Step 2 on page 42) • The *databasename* value in LDAP configuration file, **slapd.conf** (see Page 60) (TDBM only) • The *yyy* value in SQL commands to grant permissions to LDAP user (see Step 5 on page 43)
Database owner	• -UUUUUUUU- value in SPUFI script to create database, *GLDHLQ*.SGLDSAMP(TDBMDB) (see Step 2 on page 42) • The *dbuserid* value in LDAP configuration file, **slapd.conf** (see Page 60)
CLI initialization file name	• The *dsnaoini* value in LDAP configuration file, **slapd.conf** (see Page 61) • DSNAOINI DD value in JCL for LDAP server and utilities
User ID running LDAP server, **tdbm2ldif (TDBM only)**, or **ldif2tdbm (TDBM only)**	• LDAPSRV value in RACF commands to create user ID (see Page 37) • The *xxx* value in SQL commands to grant permissions to LDAP user (see Step 5 on page 43) • LDAPSRV value in RACF commands to create SSL/TLS key ring (see Page 48) • LDAPSRV value in RACF commands to create started task (see Page 101)
Server name	• DATA SOURCE value in CLI initialization file, DSNAOINI (see Step 4 on page 14) • The *servername* value in LDAP configuration file, **slapd.conf** (see Page 74)
SDBM Backend	
User ID running LDAP server or utilities	• LDAPSRV value in RACF commands to create user ID (see Page 37) • LDAPSRV value in RACF commands to create SSL/TLS key ring (see Page 48) • LDAPSRV value in RACF commands to create started task (see Page 101)

Setting the time zone

The LDAP server uses time values returned by the operating system when it records server activity or when it generates LDAP trace records. The LDAP server assumes that time values are in Universal Time Coordinated (UTC) format. The UTC time value is mapped to a (local) time zone value as specified by the **TZ** environment variable. By default, **TZ** is set to GMT0.

If you have not already done this, copy **/usr/lpp/ldap/slapd.envvars** to the **/etc/ldap** directory. Edit **/etc/ldap/slapd.envvars**, uncomment the **TZ** environment variable and set the value as desired. For more information about time zones, see Chapter 7.4, Customizing a Time Zone, in the *z/OS C/C++ Programming Guide*.

When started, the LDAP server will read an environment variable file. The default file is **/etc/ldap/slapd.envvars**. This default can be changed by setting the environment variable **LDAP_SLAPD_ENVVARS_FILE** to the full path name of the desired environment variable file. LDAP server timestamps are then generated using a UTC value and the time zone value.

Chapter 6. Setting up the user ID and security for the LDAP server

This chapter discusses how to configure and set up the products needed by your LDAP server.

If you plan to use:	You must:	See:
LDAP server as a started task	Define the LDAP server user ID.	"Setting up a user ID for your LDAP server" below and continue through remaining sections of this chapter.
LDAP server from z/OS UNIX System Services shell	Set up user ID and environment.	"Requirements for a user ID that runs the LDAP server" and continue through remaining sections of this chapter.

Setting up a user ID for your LDAP server

When running the LDAP server as a started task, it is recommended that a separate user ID be established for the LDAP server. This section describes how to define the user ID that runs the LDAP server.

In this section, some of the examples and descriptions reflect assumptions that may not apply to your environment. Following are descriptions of these assumptions, with guidance on how to use this information if they do not apply to your environment:

- Some examples use Resource Access Control Facility (RACF). You can use any z/OS external security manager that has equivalent support. You must substitute the appropriate procedures for any examples that use RACF.
- The default name **/usr/lpp/ldap** is used for the directory in which you installed the LDAP server product. If you used a different name, substitute that name in the examples and descriptions where applicable.
- The language setting **En_US.IBM-1047** is used for the locale in which you are running the LDAP server. This setting is used in the names of several directories that are referred to in this information. If you are using a different language setting, substitute that setting in the examples and descriptions where applicable. You must also specify this setting as the value of the **LANG** parameter in the environment variable file as described in Chapter 12, "Internationalization support," on page 157. The default environment variable file already sets **LANG** to **En_US.IBM-1047**.
- The name **LDAPSRV** is used for the user ID that runs the LDAP server. If you use a different name, substitute that name in the examples and descriptions where applicable.
- The name of the production directory is **/etc/ldap**. If you use a different name, you must symbolic link the names of the appropriate files in your directory to the **/etc/ldap** directory.

Requirements for a user ID that runs the LDAP server

Any user ID can be used to run the LDAP server. The examples in this chapter use a user ID of LDAPSRV in the commands provided.

The user ID that runs the LDAP server must have the following attributes:

- If you defined the BPX.DAEMON profile in the FACILITY class, the user ID must have read access to the profile.
- If you defined the BPX.SERVER profile in the FACILITY class, the user ID must have update access to the profile.
- The user ID must have read access to the data sets defined in the startup procedure.

Note that if the UID of the user ID running the LDAP server is not zero, all console messages produced by the LDAP Server are accompanied by a BPXM023I message identifying the user writing to the console.

The user ID performing the RACF commands in the following examples requires RACF SPECIAL authority.

You can use the RACF commands in the following example for the user ID that will run the LDAP server.

```
ADDGROUP LDAPGRP SUPGROUP(SYS1) OMVS(GID(2))
ADDUSER LDAPSRV NOPASSWORD DFLTGRP(LDAPGRP) OMVS(UID(1) PROGRAM ('/bin/sh'))
```

The following commands should only be entered if these facility classes are defined.

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(LDAPSRV) ACCESS(READ)
PERMIT BPX.SERVER CLASS(FACILITY) ID(LDAPSRV) ACCESS(UPDATE)
```

If you are going to set up more than one LDAP server on the same system, a separate user ID should be used for each one.

Additional setup when using SDBM

If you plan to use an SDBM backend, the following RACF commands must be entered to set up the user ID that will run the LDAP server:

```
RDEFINE FACILITY IRR.RUSERMAP UACC(NONE)
PERMIT IRR.RUSERMAP CLASS(FACILITY) ID(LDAPSRV) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

The SDBM backend also supports the RACF functions that search for users and groups with a given UID or GID value, control sharing user UID and group GID values, and retrieve a user password envelope. Usage of these functions requires additional RACF configuration and profiles, as described in the RACF documentation.

Additional setup for RACF PROXY segment and SDBM

The SDBM backend supports the PROXY segment within the RACF user profile. If you intend to use SDBM to set the BINDPW value in the PROXY segment, RACF requires that the KEYSMSTR class profile LDAP.BINDPW.KEY be created with the SSIGNON segment.

- To create the LDAP.BINDPW.KEY profile in the KEYSMSTR class, use the KEYMASKED sub-operand if no cryptographic product is installed on your system:

  ```
  RDEFINE KEYSMSTR LDAP.BINDPW.KEY SSIGNON(KEYMASKED(key-value))
  ```

  Or, use the KEYENCRYPTED sub-operand if a cryptographic product is installed:

  ```
  RDEFINE KEYSMSTR LDAP.BINDPW.KEY SSIGNON(KEYENCRYPTED(key-value))
  ```

  key-value is a Secured Sign-on Application key and must be specified as a string of 16 hexidecimal characters.
- Then, activate the KEYSMSTR class:

  ```
  SETROPTS CLASSACT(KEYSMSTR)
  ```

See *z/OS Security Server RACF Command Language Reference* for details on using these RACF commands and *z/OS Security Server RACF Security Administrator's Guide* for information on creating and using profiles.

Defining the Kerberos identity

If you plan to enable Kerberos support you need to associate a Kerberos identity with the server's user ID. The following command must be entered:

```
ALTUSER LDAPSRV PASSWORD(password) NOEXPIRED KERB(KERBNAME(ldap_prefix/hostname)
```

If the server is being run as a started task, also enter the following command:

```
ALTUSER LDAPSRV NOPASSWORD
```

Note that the *ldap_prefix* must be either ″LDAP″ or ″ldap″. Also, the *hostname* needs to be the primary hostname for the system in DNS.

If the LDAP server is located on the same machine as the Key Distribution Center then a keytab file is not necessary to start the LDAP server. However the user ID that starts the server must have at least read access to **IRR.RUSERMAP** in the **FACILITY** class. This can be done by issuing the following commands:

```
RDEFINE FACILITY IRR.RUSERMAP UACC(NONE)
PERMIT IRR.RUSERMAP CLASS(FACILITY) ID(LDAPSRV) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Protecting the environment for the LDAP server

The PDS which contains the LDAP server, SYS1.SIEALNKE, and the PDSs containing all the DLLs that the LDAP server loads must be APF-authorized to allow the LDAP server to make the necesary program control threading calls. For example:

```
SETPROG APF,ADD,DSN=SYS1.SIEALNKE,VOL=volid
```

Additionally, if program control is active on your system, the PDS which contains the LDAP server and DLLs, the PDS that contains the C runtime libraries, SYS1.LINKLIB, and SYS1.CSSLIB must be program controlled. Also, the PDS containing the DB2 CLI (Call Level Interface), *DB2HLQ*.SDSNLOAD, must be APF-authorized and program controlled. Also, if a decision has been made to use password encryption support in the TDBM backend, the OCSF DLLs must also be program controlled. See the configuration information in *z/OS Open Cryptographic Services Facility Application Programming* for more information.

Protecting the environment for SSL/TLS

If you are using SSL/TLS to secure your LDAP server, the **STEPLIB** or **LINKLIST** must include SYS1.SIEALNKE. Also, SYS1.SIEALNKE must be APF-authorized.

Chapter 7. Preparing the backends, SSL/TLS, and password encryption

This chapter discusses what you need to do to prepare the backends, SSL/TLS, and password encryption.

If you plan to use:	You must:	See:
A backend, such as TDBM, GDBM, or SDBM	Copy the configuration files.	"Copying the configuration files"
The sample server to set up a DB2 database	Use the set of example files shipped with the code.	"Creating a DB2 database for the sample server" below
TDBM or GDBM backend	Create the DB2 database and table spaces using SPUFI.	"Creating the DB2 database and table spaces for TDBM or GDBM" on page 42
SDBM backend	Set up your configuration files.	"Setting up for SDBM" on page 44
GDBM backend	Set up your configuration files.	"Setting up for GDBM" on page 45
EXOP backend	Set up your configuration files.	"Setting up for extended operations" on page 46
SSL/TLS	Enable SSL/TLS support.	"Setting up for SSL/TLS" on page 46
Password encryption	Determine the type of encryption and set up the configuration.	"Configuring for user password encryption" on page 50

Copying the configuration files

The configuration files need to be copied from the directory in which they are installed, **/usr/lpp/ldap/etc**, to the directory where they are used, **/etc/ldap**. Do not modify these files in the install directory because any service to the files will overwrite the modifications. Instead, modify them in **/etc/ldap**. The following commands copy the configuration files:

```
cp /usr/lpp/ldap/etc/slapd.* /etc/ldap/.
```

The **cp** command creates a copy of the **/usr/lpp/ldap/etc** files into the **/etc/ldap** directory. The individual files can now be modified using **oedit** or **vi**.

Creating a DB2 database for the sample server

There is a set of example files shipped in **/usr/lpp/ldap/examples/sample_server** that can be used to understand how to configure and run the LDAP server using a TDBM backend. The **README** provides step-by-step instructions for getting an LDAP server configured and started quickly. The following list shows the files shipped in that directory.

- **README** (Installation information for sample server)
- **sample.ldif** (Sample directory entry for sample server)

Creating the DB2 database and table spaces for TDBM or GDBM

When using TDBM or GDBM, the LDAP server DB2 database must be created by running two SPUFI (SQL Processor Using File Input) scripts from DB2 Interactive (DB2I). The same scripts are used for both TDBM and GDBM. DB2I is a DB2 facility that provides for the running of SQL statements, DB2 (operator) commands, and utility invocation. For details on how to use DB2I and SPUFI, see *DB2 Application Programming and SQL Guide*. Sample DB2I SPUFI scripts to create the LDAP server DB2 database are provided. To use them, do the following:

1. **Copy the SPUFI scripts over to your SPUFI input data set.**

   The SPUFI script for creating the database and table spaces can be found in *GLDHLQ*.SGLDSAMP(TDBMDB) and the script for creating the table indexes can be found in *GLDHLQ*.SGLDSAMP(TDBMINDX). (*GLDHLQ* refers to the high-level qualifier that was used to install the LDAP server data sets.)

2. **Determine values for SPUFI script.**

   In order to create the DB2 database and table spaces for TDBM or GDBM, you must first decide on certain values within these SPUFI files, as shown in Table 5. (Table 4 on page 34 lists variables that are used in more than one file or configuration step. Be sure to specify the same values where necessary.) The SPUFI scripts provide specific instructions and information to help you determine the values to use in the table. “The TDBMDB SPUFI file” on page 451 and “The TDBMINDX SPUFI file” on page 456 show examples of the files to edit and run in the SPUFI facility.

Table 5. TDBM value overview

Attribute	Value	Suggested value	Variable name in SPUFI script
Database information for TDBMDB and TDBMINDX members			
Database name		LDAPSRV	-DDDDDDDD-
Database owner		LDAPSRV	-UUUUUUUU-
Table space definitions for TDBMDB member			
Entry table space name		ENTRYTS	-AAAAAAAA-
Buffer pool name for the LDAP entry table space		BP0	-BBBB-
Long entry table space name		LENTRYTS	-CCCCCCCC-
Buffer pool name for the LDAP long entry		BP0	-DDDD-
Long attribute table space name		LATTRTS	-EEEEEEEE-
Buffer pool name for the LDAP long attribute		BP0	-FFFF-
Miscellaneous table space name		MISCTS	-GGGGGGGG-
Search table space name		SEARCHTS	HHHHHHHH-
Buffer pool name for the LDAP search table		BP0	-IIII-
Replica table space name		REPTS	-JJJJJJJJ-
Descendants table space name		DESCTS	-KKKKKKKK-
Storage group		SYSDEFLT	-SSSSSSSS-
Search column truncation size (VALUE in DIR_SEARCH)		32	-TTTT-
DN truncation size (DN_TRUNC in DIR_ENTRY)		32	-MMMM-
Maximum size of a DN (DN in DIR_ENTRY)		512	-NNNN-

3. **Modify the scripts.**

   Use the values from Table 5 on page 42 to modify the scripts. You must have a unique database name and owner for each database you are creating.

4. **Run the scripts from DB2I SPUFI.**

   Use the DB2 SPUFI (SQL Processor Using File Input) facility to create the database and table spaces.

   Be sure to run the two scripts that were copied and modified in the previous steps under a user ID with DB2 **SYSADM** authority. When the scripts complete running, scan the output data set to ensure that they ran successfully.

5. **Grant appropriate DB2 resource authorizations.**

   In order to run the LDAP server and server utilities, certain minimum DB2 resource authorizations must be granted to the user ID or user IDs that will be running these programs. Following are the suggested minimums which should be granted to those user IDs, where *xxx* is the user ID running the LDAP server or LDAP server utility, *yyy* is the database name identified in the **slapd.conf** (for TDBM only) and SPUFI file for the **databasename** option and *zzz* is the CLI plan name as specified in your DB2 CLI initialization file. Run the following statements through SPUFI (DB2 Interactive):

   ```
   grant execute on plan zzz to xxx;
   grant dbadm on database yyy to xxx;
   ```

   These privileges may be granted by any user ID with **SYSADM** authority. The commands above can be run using the DB2 SPUFI facility.

   The LDAP server requires **SELECT** access to the SYSIBM.SYSCOLUMNS table in DB2. If **SELECT** access to this table is tightly controlled in your DB2 installation, then it may be necessary to grant this access to the user ID under which the LDAP server runs by performing the following operation (either through SPUFI or another means of issuing SQL commands):

   ```
   grant select on sysibm.syscolumns to xxx;
   ```

   where *xxx* is the user ID under which the LDAP server runs. If this authority is not granted to the user ID under which the LDAP Server runs, the LDAP server will fail during start-up with an SQL -551 return code.

Partitioning DB2 tables for TDBM

If you are creating a large directory, you should partition the ″entry table space″ and ″search table space″ to improve performance and ease maintainability of the database. The following information identifies the partitioning indexes and values to use when partitioning these table spaces.

Table 6. TDBM table space partitioning indexes and values

Table space	Table name	Partitioning index	Partitioning column	Value range of column
Search tablespace	DIR_SEARCH	DIR_SEARCHX2	EID	0-999999999999999
Entry tablespace	DIR_ENTRY	DIR_ENTRYX0	EID	0-999999999999999

The EID value generated by the TDBM backend is a random 15 digit decimal number between 1 and 999999999999999. To determine the maximum value to assign to each partition, use the following formula:

```
eids_per_partition = 1000000000000000 / number_of_partitions
partition_value = eids_per_partition * partition_number
```

You can also partition the following additional table spaces in TDBM using the EID range as the partitioning value

Table 7. TDBM table space partitioning using EID range

Table space	Table name	Partitioning index	Partitioning column	Value range of column
Descendants table space	DIR_DESC	DIR_DESCX1	DEID	0-999999999999999
Long attribute table space	DIR_LONGATTR	DIR_LONGATTRX1	EID	0-999999999999999
Long entry table space	DIR_LONGENTRY	DIR_LONGENTRYX1	EID	0-999999999999999

Partitioning example

If your directory is to contain 10 million entries and you want 1 million entries per partition, the maximum value for each partition is:

Partition number	Partition maximum value
1	100000000000000
2	200000000000000
3	300000000000000
4	400000000000000
5	500000000000000
6	600000000000000
7	700000000000000
8	800000000000000
9	900000000000000
10	999999999999999

Copying a TDBM database

If you want to copy an existing TDBM database to a new TDBM database, you should use **tdbm2ldif** to unload the existing TDBM database and **ldif2tdbm** to load the results into the new TDBM database.

Note: You need to unload the existing schema separately from the rest of the database entries, but you can load both the schema and the other database entries at the same time.

See Chapter 11, “Running and using the LDAP backend utilities,” on page 135 for more information on **tdbm2ldif** and **ldif2tdbm**.

You can also use DB2 utilities to unload the existing database and load the new database.

Note: There are several table spaces, such as the miscellaneous table space (MISCTS) and the replica table space (REPTS), which contain multiple tables. You must use the appropriate options on the DB2 utilities when processing these table spaces.

Setting up for SDBM

The LDAP server can provide LDAP access to the user and group information stored in RACF. See Chapter 16, “Accessing RACF information,” on page 213 for details about how you can use this RACF information. When creating change log records for changes to RACF data, SDBM is required.

In order to configure your LDAP server to run with the SDBM backend of the LDAP server:

- If you have not already done this, copy the configuration files from the **/usr/lpp/ldap/etc** directory to the **/etc/ldap** directory (see “Copying the configuration files” on page 41).
- You need to use the following lines in your **slapd.conf** file:

```
database sdbm GLDBSDBM
suffix "your_suffix"
```

  where *your_suffix* is any valid DN (distinguished name). Be sure to provide a meaningful value for the suffix. Note that it is no longer required that the **sysplex** attribute be present in the suffix. For example, a valid suffix line is:

```
suffix "cn=RACFA,o=IBM,c=US"
```

  An SDBM suffix should not contain an alias name for an attribute. For example, the suffix cannot use the **surName** attribute (it can use the **sn** attribute instead). Also, the suffix can contain a case-sensitive attribute, but SDBM ignores case when processing the suffix.

Notes:

1. Only one SDBM backend can be defined in any given LDAP server.
2. SDBM contains an internal schema that it uses to check entries that it is adding. The schema cannot be modified.

Running SDBM with other backends

The following table gives you information on running SDBM alone or with other backends.

Table 8. SDBM with other backends

Backend	Description
SDBM with TDBM	The TDBM schema will be used for all initial DN normalization if TDBM is configured. DN normalization is performed by the server to aid in selecting the appropriate backend. All attribute types that might appear in a RACF-style DN must be defined to the TDBM schema. When starting TDBM and SDBM together, ensure that the attribute types in the SDBM suffix are also present in the TDBM schema. Examine the schema LDIF files shipped with the LDAP server to determine which schema must be loaded into TDBM.
SDBM only or SDBM with GDBM	If you are running SDBM without TDBM, be sure to comment out the TDBM database definitions in the **slapd.conf** file. Prefix each line to comment with a # (pound sign). When running without TDBM, replication and referrals are not supported.

Setting up for GDBM

The LDAP server can provide a change log containing information about changes to RACF users, TDBM entries, and some change log entries.

In order to configure your LDAP server to run with the GDBM backend of the LDAP server:

- If you have not already done this, copy the configuration files from the **/usr/lpp/ldap/etc** directory to the **/etc/ldap** directory (see *Copying the configuration files* in this chapter).
- You need to use the following lines in your **slapd.conf** file:

```
database gdbm GLDBGDBM
dbuserid userid
servername string
```

  See Chapter 25, “Change logging,” on page 307 for additional configuration options that can be specified.

- If you intend to create change log entries for changes to RACF data, you must also configure an SDBM backend and enable the LDAP Program Callable support. See *Setting up for SDBM*, in the previous section, for more information and Chapter 25, “Change logging,” on page 307, *Additional required configuration*.

Notes:

1. Only one GDBM backend can be defined in any given LDAP server.
2. GDBM contains an internal schema that it uses to define entries that it is adding. The schema can be modified.

Running GDBM with other backends

The following table gives you information on running GDBM alone or with other backends.

Table 9. GDBM with other backends

Backend	Description
GDBM with TDBM	The TDBM schema will be used for all initial DN normalization if TDBM is configured. DN normalization is performed by the server to aid in selecting the appropriate backend. All attribute types that might appear in a RACF-style DN must be defined to the TDBM schema.
GDBM only or GDBM with SDBM	If you are running GDBM without TDBM, be sure to comment out the TDBM database definitions in the **slapd.conf** file. Prefix each line to comment with a # (pound sign). When running without TDBM, replication and referrals are not supported.

Setting up for extended operations

The LDAP server supports extended operations (EXOP backend) that retrieve Policy Director data. See Chapter 20, “Using extended operations to access Policy Director data,” on page 255 for details on using this extended operations support.

To configure your LDAP server to run with the EXOP backend:

- Make sure your **slapd.conf** configuration file contains the following line for Policy Director:

  ```
  listen ldap://:pc
  ```

- Make sure your **slapd.conf** configuration file contains the following line:

  ```
  database exop GLDXPDIR
  ```

More information about the configuration file and options is in Chapter 8, “Customizing the LDAP server configuration,” on page 53.

Setting up for SSL/TLS

The LDAP server contains the ability to protect LDAP access with Secure Sockets Layer (SSL) and Transport Layer (TLS) security. There are two types of connections that support secure communication:

- An SSL/TLS only secure connection. This connection requires that the first communication between the client and the server be the handshake that negotiates the secure communication. From that point on only secure communication can occur on the connection.
- A bimodal connection that supports secure and non-secure communication. The client is expected to begin communication in a non-secure mode. At some time during communication, the client may change to secure communication by sending a StartTLS extended operation after which the handshake to negotiate secure communication occurs followed by secure communication. The client may shutdown secure communication causing a StopTLS alert to be sent and the server will continue communication in a non-secure mode. At a later time, the client may restart secure communication by sending another StartTLS extended operation followed by the handshake.

Both types of connections require that System SSL be configured for use by the LDAP Server.

Using SSL/TLS protected communications

The Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols use public-key infrastructure (PKI) algorithms to establish and maintain an encrypted communications path between a client and server. In z/OS, the ability to set up and communicate over SSL/TLS protected communication links is provided by the LDAP server with a set of services provided in z/OS (the z/OS Cryptographic Services System SSL set of services).

In order for the LDAP client to communicate with an LDAP server over an SSL/TLS-protected TCP/IP socket connection, the LDAP server must transmit a certificate to the LDAP client and, optionally, the client can transmit its certificate to the LDAP server. The LDAP client and server must verify that the certificates they received are valid. Once the LDAP client and LDAP server have determined the validity of the certificates provided to them, SSL/TLS-protected communication occurs between the LDAP client and server.

The LDAP client and server verify the certificates sent to them by using public-key digital signatures. The LDAP client and server take the certificates and compare the digital signature in the certificates with a signature that it computes based on having the public-key of the signer of the certificate. In order to do this, the LDAP client and server must have the public-key of the signer of the certificates. The LDAP client and server obtain this by reading a file that contains these public-keys. This file is called a key database.

A key database, or RACF key ring, contains the public-keys that are associated with signers of certificates. These public-keys are, in reality, contained in certificates themselves. Thus, verifying one certificate requires the use of a different certificate, the signer's certificate. In this fashion, a chain of certificates is established, with one certificate being verified by using another certificate and that certificate being verified by yet another certificate, and so on. A certificate, and its associated public key, can be defined as a *root* certificate. A root certificate is *self-signed*, meaning that the public-key contained in the certificate is used to sign the certificate. Using a root certificate implies that the user *trusts* the root certificate.

The key databases, or key rings, used by the LDAP client and server must contain enough certificates in order to verify the certificates sent by the LDAP client and server during the start-up of the SSL/TLS connection. If either certificate is self-signed, then that certificate must be stored in the other's key database. If the certificates are signed by some other certificate signer, then the signer's certificate and any certificates that this certificate depends upon must be stored in the key databases. The key databases used by the LDAP client and server must also contain the certificates that they will transmit to each other during the startup of the SSL/TLS-protected communications.

Enabling SSL/TLS support

The following high-level steps are required to enable SSL/TLS support for LDAP. These steps assume you have already installed and configured the LDAP directory server and installed z/OS Cryptographic Services System SSL. The datasets containing the LDAP and SSL code must be APF authorized and available to the LDAP server.

1. Generate the LDAP server private key and server certificate and mark it as the default in the key database or use its label on the **sslCertificate** configuration file option (see "Using SSL/TLS protected communications").
2. Configure the LDAP server to listen for LDAP requests and configure the type of authentication wanted, server and optionally client authentication (see "Setting up the security options for the LDAP server" on page 48).
   - For a secure only socket, a **listen** configuration or command line option must be set up for the secure port.
   - For a bimodal socket, a **listen** configuration or command line option must be set up for the non-secure port.
3. Restart the LDAP server.

Creating and using a key database or key ring

The LDAP client and server use the System SSL functions provided in z/OS to set up SSL/TLS protected communications. The System SSL capability requires a key database or key ring to be set up before SSL/TLS protected communications can begin.

The key database is a password protected file stored in the hierachical file system (HFS). This file is created and managed using a utility program provided with System SSL called **gskkyman**. Directions for using the **gskkyman** utility can be found in z/OS: *System Secure Sockets Layer Programming*. The key database file that is created must be accessible by the LDAP server.

The key ring is maintained by RACF. This object is created and managed using the RACF Digital Certificate command, **RACDCERT**. Directions for using the **RACDCERT** command can be found in *z/OS Security Server RACF Command Language Reference*.

The user ID under which the LDAP server runs must be authorized by RACF to use RACF key rings. To authorize the LDAP server, you can use the RACF commands in the following example:

```
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(LDAPSRV) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(LDAPSRV) ACCESS(CONTROL)
```

Remember to refresh RACF after doing the authorizations.

```
SETROPTS RACLIST(FACILITY) REFRESH
```

For testing purposes, the LDAP server can use a self-signed certificate. In this case, the certificate of the LDAP server must also be stored in the key database, or key ring, of the LDAP client in order for SSL/TLS protected LDAP communications to work between the client and server.

The certificate that the LDAP server is going to use should be stored as the default certificate in a key database or key ring, or the certificate label of the certificate is configured in the LDAP server using the **sslCertificate** option.

Obtaining a certificate

The LDAP server or client can obtain a certificate by contacting a certificate authority (CA) and requesting a certificate. Utilities to formulate a certificate request are provided by System SSL, **gskkyman**, and RACF, **RACDCERT**. This certificate request is usually passed to the CA by means of an electronic mail message or by an HTML form which is filled out using a web browser. Once the CA verifies the information for the LDAP client or server, a certificate is returned to the requester, usually by an electronic mail message. The contents of the mail message are used to define the certificate in the key database or key ring.

Setting up the security options for the LDAP server

The following options for SSL/TLS can be set in the **slapd.conf** file. They are described in detail in “Configuration file options” on page 56.

- **listen**
- **sslAuth**
- **sslCertificate**
- **sslCipherSpecs**
- **sslKeyRingFile**
- **sslKeyRingFilePW**
- **sslKeyRingPWStashFile**

Notes:

1. The **replKeyRingFile** and **replKeyRingPW** options are no longer necessary or recognized by the LDAP server. These options should be removed from the configuration file.

2. The **security**, **port**, and **securePort** options have been deprecated by the **listen** option. For more information, see the **listen** option on page 65.

LDAP can be configured for SSL/TLS in two ways:

- For secure only communication, specify one or more **listen** options for secure communications in the following format:

  `ldaps://[IP_address | hostname] [:portNumber]`
- For bimodal (non-secure/secure) communication, specify one or more **listen** options for non-secure communications in the following format:

  `ldap://[IP_address | hostname] [:portNumber]`

For more information on the **listen** option, refer to page 65.

sslKeyRingFile specifies the name of the key database or the key ring used by the LDAP server. This key database or key ring is also used for SSL/TLS protected replication. Because the replicating server may be acting as both a replica server and an LDAP server, the replica server's certificate (or CA's certificate) must be contained in the replicating server's key database file or key ring.

A key database requires a password. The password may be specified on the **sslKeyRingFilePW** option or the name of a password stash file may be specified on the **sslKeyRingPWStashFile** option in the configuration file. Use of a stash file provides a method of specifying a password in a form that can not be easily read by a human. The **gskkyman** utility provides a function to create the key database password stash file.

When a RACF key ring is used instead of a key database, the **sslKeyRingFilePW** and **sslKeyRingPWStashFile** should not be specified in the configuration file.

The LDAP server is configured to provide server and, optionally, client authentication. The **sslAuth** option controls this setting.

With server authentication, the LDAP server must have a digital certificate (based on the X.509 standard). This digital certificate is used to authenticate the LDAP server to the LDAP client application. The LDAP server supplies the client with the LDAP server's X.509 certificate during the initial SSL handshake. If the client validates the server's certificate, then a secure, encrypted communication channel is established between the LDAP server and the LDAP client application.

In addition, if the LDAP server is configured to use server and client authentication, and the client sends a digital certificate on the initial SSL handshake, it must be validated by the LDAP server before the secure encrypted communication channel is established between them.

Client authentication by the LDAP server facilitates the use of certificate bind (SASL mechanism of **EXTERNAL**) by an LDAP client. The bind identity becomes the distinguished name in the client digital certificate.

Note: If the LDAP server is configured for both server and client authentication, but a client does not send a digital certificate, then the server will act as if configured for server authentication only. This provides backward compatibility of the LDAP server.

The **sslCertificate** option indicates the label of the server certificate that is to be used. If the default certificate has not been set in the key database or key ring, or if a certificate other than the default ceritficate is desired then this option is needed.

The **sslCipherSpecs** option specifies the cipher specifications that will be accepted from clients. If this option is not specified then all cipher specifications supported by the LDAP server will be used. Depending upon the level of System SSL support, the list of acceptable cipher specifications may be lowered because certain specifications may not be supported by System SSL for that level of the product.

Setting up an LDAP client

As with the LDAP server, the LDAP client that wishes to use SSL/TLS protected communication needs access to a key database or key ring. If the LDAP server you are going to contact is using a self-signed certificate (as is done frequently while testing SSL/TLS protected communications between an LDAP client and server), then the self-signed certificate of the LDAP server must be stored into the LDAP clients key database or key ring.

If the LDAP server you are going to contact is using a certificate which is signed by a certificate authority (CA), you must ensure that the certificate for the CA is contained in the key database. Use whatever means is provided by the CA for obtaining the CA certificate. The certificate should be obtainable in a format that is acceptable to the **gskkyman** utiltity or **RACDCERT** command.

If the LDAP server is configured for server and client authentication and the client wants client authentication to occur, then the LDAP client must obtain its own certificate from a CA and store it in the clients own key ring or key database and mark it as the default.

Once the key database file or key ring is created and contains the proper certificates, then the LDAP client is ready to perform SSL/TLS protected communications with an LDAP server. The LDAP Operation Utilities (for example, **ldapsearch**) can be used to communicate securely with the LDAP server using a secure only connection. The utilities are explained in *z/OS Integrated Security Services LDAP Client Programming*.

Using LDAP client APIs to access LDAP using SSL/TLS

The **ldap_ssl_client_init** and **ldap_ssl_init** APIs can be used to start a secure only connection to an LDAP server. A description of these APIs can be found in *z/OS Integrated Security Services LDAP Client Programming*.

Support of certificate bind

The SASL bind mechanism of EXTERNAL is supported by the LDAP server. This means that the authentication on the bind is performed using the data obtained during the SSL/TLS client authentication that was performed on the SSL/TLS handshake with the client.

To use SASL External bind, the following steps must occur:

- The LDAP server must be configured and started with **sslAuth** set to **serverClientAuth** so that the server can authenticate the client.
- The client connects to the LDAP server and performs the SSL/TLS handshake. The handshake sends the client certificate to the LDAP server.
- The client performs a SASL bind with the mechanism of **EXTERNAL**.

At this point, the LDAP server will consider the bind DN of the client for authorization purposes to be the client's DN as transmitted in the client's certificate on the handshake.

Configuring for user password encryption

The LDAP server allows prevention of unauthorized access to user passwords in the TDBM backend. The **userPassword** attribute values can be encoded when stored in the directory, which prevents clear text passwords from being accessed by any users, including the system administrators. In the current implementation, only the **userPassword** attribute values are encrypted. Use of the terms "user password" and "password" refer to the **userPassword** attribute. Use of the term "user entry" refers to an entry in TDBM that contains a **userPassword** attribute.

Note: The z/OS LDAP server does not allow **userPassword** attributes in distinguished names.

The administrator may configure the server to encode **userPassword** attribute values in either a one-way hash format or a two-way, symmetric, encryption format.

After the server is configured and started, any new passwords for new user entries, or modified passwords for existing user entries are encoded before they are stored in the TDBM backend. The encoded passwords are tagged with the encoding algorithm name so that passwords encoded in different formats can coexist in the directory. When the encoding configuration is changed, existing encoded passwords remain unchanged and continue to be usable.

Note about password encryption and replication: Some important considerations are described in "Password encryption and replication" on page 290.

When **ldif2tdbm** is used to load a TDBM backend, all clear text user passwords in new entries are encrypted by the method specified in the configuration file.

If there are encrypted **userPassword** values in the LDAP database, the unload utility **tdbm2ldif** unloads the TDBM backend to LDIF format with the password in the binary format of:

```
userPassword::  base64encoded_and_tag_encryptedvalue
```

This format can be loaded by the load utility, for example at a replica server, and preserves the encrypted passwords.

The **-t** option on **tdbm2ldif** unloads the user passwords in an encrypted format that might be more appropriate for loading by non-z/OS LDAP servers. The LDIF format unloaded by the **-t** option can be called encryption ″tag visible″ format and looks like:

```
userPassword: {tag}base64encoded_and_encryptedvalue
```

where *tag* is **none**, **crypt**, **MD5**, **SHA**, or **DES:***keylabel*.

In this format the tag is visible, and only the password itself is encrypted and base64 encoded.

Notes:

1. The tag is enclosed in a left brace and a right brace. One colon is used between the **userPassword** keyword and the value, as opposed to two colons in the standard LDIF format of **userPassword** unloaded by **tdbm2ldif**. The format produced by **tdbm2ldif -t** cannot be read by the z/OS **ldif2tdbm** utility programs. It is intended for other LDAP platforms or tools that may be able to interpret this LDIF format with the encryption tag visible.
2. The values returned by the crypt() algorithm are not portable to other X/Open-conformant systems. This means that user password values encoded by the crypt() algorithm and unloaded as tagged output using **tdbm2ldif -t** are not portable when loaded by another platform's load utility.

If the TDBM backend is already loaded and the LDAP server is running, the **db2pwden** utility is provided to encrypt all clear text **userPassword** attribute values in the method configured on the LDAP server.

The **db2pwden** utility is similar to the LDAP operation utilities, such as **ldapsearch**, in that it acts like a client to the LDAP server and has similar command line options. See "db2pwden utility" on page 154 for information about the **db2pwden** utility and see *z/OS Integrated Security Services LDAP Client Programming* for more information about LDAP operation utilities, such as **ldapsearch**. The **db2pwden** utility must be run by the LDAP server administrator using the **adminDN** and password configured on the server.

Be aware that once a password is encrypted in a one-way hash, its clear text value can no longer be retrieved or displayed.

- **One-way hash formats:**
  crypt

MD5
SHA

A crypt, MD5, or SHA hashed password can be used for password matching on an LDAP simple bind, but it cannot be decrypted. During simple bind, the bind password is hashed and compared with the stored **userPassword** attribute values for matching verification.

MD5 and SHA hashing require the z/OS Open Cryptographic Services Facility (OCSF) be installed and configured, and the necessary security authorizations to be set up for the LDAP server user ID in RACF. See the configuring information in *z/OS Open Cryptographic Services Facility Application Programming* for instructions on how to do this.

For applications which require retrieval of clear passwords, such as middle-tier authentication agents, the directory administrator must configure the LDAP server to perform either a two-way encoding or no encryption of user passwords. Access to password data stored in the directory can be protected by the access control mechanism of the directory.

- **Two-way encryption format:**
  DES

  The DES algorithm is provided to allow values of the **userPassword** attribute to be encoded in the TDBM backend and retrieved as part of an entry in the original clear format. Some applications such as middle-tier authentication servers require passwords to be retrieved in clear text, however, corporate security policies might prohibit storing clear passwords in a secondary permanent storage. This option satisfies both requirements.

  A DES encrypted password can be used for password matching on a simple bind and can be decrypted to be returned as clear text on a search request when the client is authorized to do so. During simple bind, the bind password is encrypted and compared with the stored version for matching verification. During a search, if the client is authorized through directory access controls to see the **userPassword** attribute value, then it is decrypted and returned as clear text.

  DES encryption requires OCSF be installed and configured, and the necessary security authorizations to be set up for the LDAP server user ID in RACF. See the configuring information in *z/OS Open Cryptographic Services Facility Application Programming* for instructions on how to do this.

  The DES algorithm also requires a key label and the single-length data key it refers to for password matching and decryption to take place. When a **userPassword** is stored in TDBM, the key label is stored along with the tag and encrypted user password. The Integrated Cryptographic Service Facility (ICSF), Key Generator Utility Program (KGUP) and Cryptographic Key Data Set (CKDS) are used to generate and store the key label and RACF is used to limit access to this DES encryption key to only the LDAP server. See the information on managing cryptographic keys and using the Key Generator Utility Program in *z/OS Cryptographic Services ICSF Administrator's Guide* for information on generating, storing into CKDS, and authorizing a Data-Encrypting Key. Remember to refresh CKDS and RACF after entering and authorizing a key.

  The DES key label must correspond to the same DES keys across a sysplex and be accessible to all LDAP servers that are using the same TDBM backend. It is recommended that you use one DES key label. If multiple DES key labels are used by different servers in the sysplex, for example, then all the servers in the sysplex need to have access to all the keys.

A simple bind will succeed if the password provided in the bind request matches with any of the multiple values of the **userPassword** attribute. Note that depending on when **userPassword** values are stored in the directory, different attribute values can be encoded using different encoding methods.

Note: The crypt() algorithm, implemented across many platforms, accepts only the first eight characters of a password. As a result, any password supplied on an **ldap_simple_bind** operation or **ldap_compare** operation that matches the first eight characters of a **userPassword** attribute value encrypted with the crypt() algorithm in the directory will match.

Chapter 8. Customizing the LDAP server configuration

This chapter contains information on how to set up the **slapd.conf** configuration file and how to configure the LDAP server to run with the options you choose. The **slapd.conf** file is also used by the LDAP utilities.

Creating the slapd.conf file

This section discusses what is necessary for creating the **slapd.conf** configuration file. Specifically, this section:

- Describes where the **slapd.conf** file is located
- Shows the configuration file format
- Provides a checklist for the configuration file options
- Lists all of the configuration file options
- Describes how to establish the administrator DN and password and a replica server DN and password
- Discusses encryption using the **pwEncryption** option

See /usr/lpp/ldap/ldap/etc/slapd.conf for the configuration file.

Locating slapd.conf

All LDAP server runtime configuration is accomplished through the configuration file **slapd.conf**, installed in the **/usr/lpp/ldap/etc** directory. If this is your first time installing the LDAP server, create a new copy of **slapd.conf** with:

```
cp /usr/lpp/ldap/etc/slapd.conf /etc/ldap/slapd.conf
```

and edit **/etc/ldap/slapd.conf**.

An alternate configuration file can be specified through a command-line option to the LDAP server and other LDAP programs.

The initial configuration contains default versions of some configuration settings. It does not contain a database suffix.

Configuration file format

The **slapd.conf** file consists of the following sections:

Global section
: Contains configuration options that apply to the LDAP server as a whole (including all backends).

SDBM backend-specific section
: Contains configuration options that apply to the SDBM backend.

GDBM backend-specific section
: Contains configuration options that apply to the GDBM backend.

TDBM backend-specific section
: Contains configuration options that apply to the TDBM backend. It is possible to have one or more of these sections depending on how many TDBM backends your installation uses.

EXOP backend-specific section
: Contains only the **database** statement necessary for the EXOP backend.

Noted below are some rules for setting up **slapd.conf**:

- The configuration file always contains a global section followed by one or more database backend sections that contain information specific to a backend instance.
- The configuration file or files must be in code page IBM-1047.
- For single-valued options that appear more than once, the last appearance in the **slapd.conf** file is used.
- Blank lines and comment lines beginning with the pound sign character (#), in column one, are ignored.
- If a line begins with one or more blank spaces, it is considered a continuation of the previous line. Therefore, begin each configuration option in column one.
- If an argument contains one or more blank spaces, the argument should be enclosed in double quotation marks (for example, `"argument one"`). If an argument contains a double quotation mark or a backslash character (\), the double quotation mark or backslash character should be preceded by a backslash character (\).
- A pound sign (#) **cannot** be used at the end of a configuration line to denote commentary.

Figure 6 shows the general format of **slapd.conf**.

```
# Global options - these options apply to every database
<global configuration options>

# SDBM database definition and configuration options
database sdbm GLDBSDBM
<configuration options specific to SDBM backend>

# TDBM database definition and configuration options
database tdbm GLDBTDBM
<configuration options specific to TDBM backend>

# GDBM database definition and configuration options
database gdbm GLDBGDBM
<configuration options specific to GDBM backend>

# EXOP database definition and configuration options
database exop GLDXPDIR
```

Figure 6. General format of slapd.conf

Configuration file checklist

The following table is provided to assist you in determining which configuration file options you will need to use in your **slapd.conf** file. Depending on the section in the configuration file (Global, SDBM, TDBM, GDBM or EXOP), certain topics (SSL, schema, replication, and so on) have options that are required or optional.

Table 10. Configuration file options checklist

Section/topic	Check	Options
Global		**adminDN** is required **adminPW, altServer, commThreads, db2terminate, digestRealm, idleConnectionTimeout, include, listen, logfile, maxConnections, pcThreads, referral, sendV3stringsoverV2as, serverEtherAddr, sizeLimit, timeLimit,** and **validateincomingV2strings** are optional.

Table 10. Configuration file options checklist (continued)

Section/topic	Check	Options
SSL/TLS		**sslAuth, sslCertificate, sslKeyRingFilePW**, **sslKeyRingPWStashFile**, and **sslCipherSpecs** are optional **sslKeyRingFile** is required if a **listen** option is initialized for secure socket communications or a **listen** option is initialized for non-secure socket communications that is intended to support switching to secure socket communications once the connection is established.
Sysplex		**sysplexGroupName** and **sysplexServerName** are optional
Kerberos		**supportKrb5** and **serverKrbPrinc** are required **krbKeytab** and **krbLDAPAdmin** are optional
SDBM backend		**database** and **suffix** are required **include**, **readOnly**, **sizeLimit** and **timeLimit** are optional
Kerberos		**krbIdentityMap** is optional
TDBM backend		**database, databasename, dbuserid,** and **suffix** are required **aclSourceCacheSize, attrOverflowSize, changeLoggingParticipant, dnCacheSize, dnToEidCacheSize, dsnaoini, entryCacheSize, entryOwnerCacheSize, extendedGroupSearching, filterCacheBypassLimit, filterCacheSize, include, persistentSearch, readonly, schemaReplaceByValue, servername, sizeLimit,** and **timeLimit** are optional
Password encryption		**pwEncryption** is optional
Replication		**masterServer, masterServerDN, masterServerPW, peerServerDN,** and **peerServerPW** are optional
Multi-server		**multiserver** is optional
Kerberos		**krbIdentityMap** is optional
Native authentication		**useNativeAuth** and **nativeAuthSubtree** are required **nativeUpdateAllowed** is optional
GDBM backend		**database** and **dbuserid** are required **aclSourceCacheSize, attrOverflowSize, changeLogging, changeLoggingParticipant, changeLogMaxAge, changeLogMaxEntries, dnCacheSize, dnToEidCacheSize, dsnaoini, entryCacheSize, entryOwnerCacheSize, filterCacheBypassLimit, filterCacheSize, include, persistentSearch, readOnly, schemaReplaceByValue, servername, sizeLimit,** and **timeLimit** are optional
Multi-server		**multiserver** is optional
EXOP backend		**database** is required **include** is optional

Note: Be sure to specify **adminDN**. You can specify the **adminPW** here or in a database entry. See "Establishing the administrator DN and the replica server DN and passwords" on page 87 for more information. Note that the use of the **adminPW** option is strongly discouraged. Instead, an existing entry in the directory should be designated as the **adminDN**.

Configuration file options

This section contains an alphabetical listing of the configuration file options. For each option, a table shows an **X** in the areas (Global, TDBM, SDBM, GDBM, and EXOP) of the configuration file where the option can be used.

Specifying a value for filename

In the configuration file options, the value for *filename* can be specified in one of the following ways:

/pathname/filename
: Specifies the full path name of a file in the z/OS UNIX System Services Hierarchical File System (HFS).

filename
: Specifies a path name that is relative to the current working directory of the LDAP server. Note that when running from a started task or batch, there is no current working directory defined. This format is not recommended.

//'dataset.name'
: Specifies the fully-qualified name of a configuration file stored in a sequential dataset.

//'dataset.name(member)'
: Specifies the fully-qualified name of a configuration file stored in a partitioned dataset.

//DD:DDNAME
: Specifies the DDNAME of a configuration that has been specified as a DD card in the JCL for the batch job or started task.

Specifying a value for a distinguished name

The value for the following configuration options is a distinguished name (DN): **adminDN, masterServerDN, peerServerDN, nativeAuthSubtree**, and **suffix**. Special characters (as identified in RFC 2253) used in the DN must be properly escaped using two backslashes (\\). Note that the double backslashes are only needed in the configuration file; in all other usages, the special characters are usually prefixed by a single backslash. See Chapter 16, "Accessing RACF information," on page 213 for exceptions to this when using SDBM. See IETF RFC 2253 *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names* for valid DN formats.

For example, to use a RACF® userid `admin#1` as the LDAP administrator, the **adminDN** configuration option would look similar to:

```
adminDN "racfid=admin\\#1,profiletype=user,cn=myRacf"
```

aclSourceCacheSize *num-entries*

Global	TDBM	SDBM	GDBM	EXOP
	X		X	

Specifies the maximum number of entries to store in the ACL Source cache. This cache holds information regarding ACL definitions within the database. Retrieval of information from this cache avoids database read operations when resolving access permissions.

The maximum size of this value is 2147483647. A value of 0 indicates that the cache is not used.
Default = 100

adminDN *dn*

Global	TDBM	SDBM	GDBM	EXOP
X				

The distinguished name (DN) of the administrator for this LDAP server. This DN will have unrestricted access to all entries in the local directory. The name that is chosen should be descriptive of the person that will have knowledge of and administer the LDAP server. The format of the name must be in DN format which is described in Chapter 13, "Data model," on page 161. It is recommended, though not necessary, that the DN have the same suffix as one of the **suffix** option values in the configuration file.

"Establishing the administrator DN and the replica server DN and passwords" on page 87 describes how to set up your administrator DN.

With LDAP V3 support, UTF-8 characters can be used for textual attributes stored in the directory. It is also desirable to allow any UTF-8 character to appear in distinguished names, and in particular, the **adminDN** distinguished name. Because the LDAP configuration files are defined to hold information only in the IBM-1047 character set, a solution is required for the configuration files that allows you to use distinguished names containing UTF-8 characters but using only the IBM-1047 character set. To solve this problem, an escape mechanism has been introduced for purposes of entering either the **adminDN** or the **masterServerDN**. This escape mechanism allows the entry of UTF-8 characters while keeping the input string value to within the IBM-1047 character set. The escape mechanism employed requires that you express UTF-8 characters which are not within the X'00' - X'7F' range (7-bit ASCII which is the single-byte form of UTF-8 characters) in the form of a set of four character representations. This representation has the form ″&*nmm*″ where 0<*n*<3 and 0<*m*<7. You might recognize *nmm* as being an octal value for a byte of information. Thus, if you want to create an **adminDN** which was the following distinguished name:

```
cn=Peter <U umlaut>nger, o=Widgets, c=DE
```

enter the **adminDN** into this option value as:

```
cn=Peter &nmm&nmmnger, o=Widgets, c=DE
```

Because the <U umlaut> is not within the 7-bit ASCII range, the value must be escaped to the octal representation of the UTF-8 multi-byte character. In the case of <U umlaut>, the Unicode code point is X'00DC'. Converted to UTF-8, this character is a multi-byte sequence: X'C3BC'. (Refer to "UTF-8 support" on page 157 for conversion information.) Converted to the escaped form for input into the **adminDN** field, this character is represented as ″&303&234″ since X'C3' is octal 303 and X'BC' is octal 234. Thus, the **adminDN** above would be entered as:

```
cn=Peter &303&234nger, o=Widgets, c=DE
```

If there is a case where you need to enter an **adminDN** string which contains the string ″&*nmm*″ where 0<*n*<3 and 0<*m*<7, then you must escape the ampersand by using its octal representation which is ″&046″.

adminPW *string*

Global	TDBM	SDBM	GDBM	EXOP
X				

The password of the administrator (**adminDN**) for this server.

"Establishing the administrator DN and the replica server DN and passwords" on page 87 describes how to set up your administrator password.

Note: Use of the **adminPW** configuration option is strongly discouraged in production environments. Instead, specify your **adminDN** as the distinguished name of an existing entry in the directory information tree. This will eliminate passwords from the configuration file.

altServer *ldap_URL*

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies an equivalent server to this LDAP server. It may or may not be a replica, but should contain the same naming contexts. The alternate server is specified as an LDAP URL.

In the following example, `myldap.server.com` is the host name and `3389` is the port number of the LDAP URL:

```
altServer ldap://myldap.server.com:3389
```

In the following example,

```
5f1b:df00:ce3e:e200:20:800:2078:e3e3
```

is the IPv6 address and `389` is the port number of the LDAP URL:

```
altServer ldap://[5f1b:df00:ce3e:e200:20:800:2078:e3e3]:389
```

Use the **altServer** configuration option to list alternate servers in the **altServer** root DSE attribute. The **altServer** root DSE attribute is used to list other servers that may be contacted in case this server is not available at some later time.

attribute *name* **[***name2* **...***namen***] {bin | ces | cis | tel | dn}** *colname maxlen* **{normal | sensitive | critical}**

Global	TDBM	SDBM	GDBM	EXOP
X				

Note: The **attribute** option is now ignored by the LDAP server.

attrOverflowSize *num-of-bytes*

Global	TDBM	SDBM	GDBM	EXOP
	X		X	

Specifies, in bytes, the minimum size of an attribute value required to store the value in a long attribute value table. The choice of this value allows large attribute values (such as **JPEG** and **GIF** files) to be stored in a separate DB2® table in a separate DB2 table space. The value must be between 1 and 2147483647.

Default = 255

changeLogging {on | off}

Global	TDBM	SDBM	GDBM	EXOP
			X	

Turns change logging on or off.

When change logging is on, all change logging operations are allowed. When change logging is off, change log entries can be searched, modified, and deleted, but no new change log entries can be created and no automatic trimming of the change log is performed.

Default = **on**

changeLoggingParticipant {yes | no}

Global	TDBM	SDBM	GDBM	EXOP
	X		X	

Allows/disallows change logging for changes made to entries in this backend.

Note: This option does not turn on or off change logging. That is done by the **changeLogging** option.
When specified in GDBM, **changeLoggingParticipant** only prevents the logging of modifications of the change log root and schema entries, which are the only changes in GDBM that can be logged.
Default = **yes**

changeLogMaxAge *nnn*

Global	TDBM	SDBM	GDBM	EXOP
			X	

Specifies the maximum age in seconds of an entry in the change log. Change log entries are deleted when they have been in the change log longer than this value, except if **changeLogging off** is specified. The value must be between 0 and 2147483647. A value of 0 indicates that there is no maximum.
Default = 0

changeLogMaxEntries *nnn*

Global	TDBM	SDBM	GDBM	EXOP
			X	

Specifies the maximum number of entries that the change log can contain. If the number of change log entries exceeds this value, change log entries with the lowest change numbers are deleted until the number of remaining entries is 95% of the maximum, except if **changeLogging off** is specified. The value must be between 0 and 2147483647. A value of 0 indicates that there is no maximum.
Default = 0

commThreads *num-threads*

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies the number of threads to be initialized for the communication thread pool. This thread pool handles the connections between the LDAP server and its clients.
Default = 10

It is recommended that **commThreads** be set to approximately two times the number of CPUs that are running in your LPAR. However, this is a general rule depending upon the activity that your LDAP server experiences.

The **commThreads** option deprecates the **maxThreads** and **waitingThreads** options, which are no longer evaluated by the LDAP server.

database *dbtype dblibpath* **[***name***]**

Global	TDBM	SDBM	GDBM	EXOP
	X	X	X	X

Marks the beginning of a new database section.

- For *dbtype*:
  - IBM supports **tdbm** (DB2), **sdbm** (RACF), **gdbm** (DB2) and **exop** (extended operations). The type **config** is reserved by the LDAP server and should not appear as **dbtype** in your configuration files.
- For *dblibpath*:
  - This is the file name of the shared library (DLL) containing the backend database code. Unless you have changed the names of the LDAP DLLs, specify **GLDBTDBM** when *dbtype* is **tdbm**, **GLDBSDBM** when *dbtype* is **sdbm**, **GLDBGDBM** when *dbtype* is **gdbm**, and **GLDXPDIR** when *dbtype* is **exop**.
- For *name*:
  - This optional value is a name that is used to identify this backend. If specified, the *name* must be different (ignoring case) than the *name* specified on any other **database** record in this configuration file. You cannot specify **cdbm1** as the *name*.

databasename *dbname*

Global	TDBM	SDBM	GDBM	EXOP
	X			

Specifies the name of the DB2 database this backend uses to store directory data.

dbuserid *userid*

Global	TDBM	SDBM	GDBM	EXOP
	X		X	

Specifies a z/OS user ID that will be the owner of the DB2 tables.

Note: Specify different values for **dbuserid**, if you are configuring both GDBM and TDBM. These backends cannot share a database.

db2terminate {terminate | restore}

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies how the LDAP server will react to a termination of DB2.

If set to **db2terminate**, the LDAP server will shutdown.

If set to **restore**, the LDAP server will disconnect from DB2 but remain running to allow access to non-DB2 backends (for example, SDBM). When DB2 is once again active, the LDAP server will re-connect to DB2. There will be no access allowed to DB2-based backends (TDBM and GDBM) during the time when DB2 is down. Client requests to those backend are rejected with LDAP_OPERATIONS_ERROR return code and a reason code message that includes "DB2 Unavailable".

Note: **db2terminate** is ignored and no DB2 monitoring is done if no DB2-based backend (TDBM or GDBM) is configured.

If using a sysplex distributor, this configuration option should be set to **terminate**. This will allow client requests to be routed to other LDAP servers in the sysplex who can connect to their databases.

Default = **restore**

digestRealm *hostname*

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies a realm name to be used when doing DIGEST-MD5 or CRAM-MD5 SASL authentication binds to the LDAP server. The **digestRealm** is used to help calculate a hash for DIGEST-MD5 and CRAM-MD5 authentication binds. It is suggested that the *hostname* be a DNS-host name and not an IP address.

Default = fully qualified *hostname* of the LDAP server if a DNS (Domain Name Server) is active on the system. Otherwise, the default is the name of the host processor.

dnCacheSize *num-entries*

Global	TDBM	SDBM	GDBM	EXOP
	X		X	

Specifies the maximum number of entries to store in the Distinguished Name normalization cache. This cache holds information related to the mapping of Distinguished Names between their raw form and their canonical form. Retrieval of information from this cache reduces processing required to locate entries in the database.

The maximum size of this value is 2147483647. A value of 0 indicates that the cache is not used.
Default = 1000

dnToEidCacheSize *num-entries*

Global	TDBM	SDBM	GDBM	EXOP
	X		X	

Specifies the maximum number of entries to store in the Distinguished Name to Entry Identifier mapping cache. This cache holds information related to the mapping of Distinguished Names in their canonical form and their Entry Identifier within the database. Retrieval of information from this cache avoids database read operations when locating entries within the database.

If this option is specified in multi-server mode, a warning message is issued and a value of 0 is used.

The maximum size of this value is 2147483647. A value of 0 indicates that the cache is not used.
Default = 1000 in single-server mode, 0 in multi-server mode.

dsnaoini *filename*

Global	TDBM	SDBM	GDBM	EXOP
	X		X	

Specifies the name of the CLI Initialization file or sequential data set (or PDS member) you created in step 4 on page 14. If the **dsnaoini** option is set in the configuration file, the LDAP server will export the **DSNAOINI** environment variable to the value specified for the configuration option.

In addition to using the **dsnaoini** configuration option or the **DSNAOINI** environment variable, a DSNAOINI DD card can be used to specify the CLI Initialization file. If the DSNAOINI DD card is specified in the JCL for the job/started task for the LDAP server, then neither the **dsnaoini** configuration option value nor the **DSNAOINI** environment variable need to be specified. "Running the LDAP server using data sets" on page 104 gives more information on this process. See *DB2 ODBC Guide and Reference* for details on ways to specify the CLI initialization file. In order for the TDBM or GDBM backend to run, the initialization file must be specified in one of the ways indicated.

Refer to page 56 for information on specifying the *filename*.

entryCacheSize *num-entries*

Global	TDBM	SDBM	GDBM	EXOP
	X		X	

Specifies the maximum number of entries to store in the Entry cache. This cache holds information contained within individual entries in the database. Retrieval of information from this cache avoids database read operations when processing entries within the database.

If this option is specified in multi-server mode, a warning message is issued and a value of 0 is used.

The maximum size of this value is 2147483647. A value of 0 indicates that the cache is not used.
TDBM Default = 5000 in single-server mode, 0 in multi-server mode.
GDBM Default = 0.

entryOwnerCacheSize *num-entries*

Global	TDBM	SDBM	GDBM	EXOP
	X		X	

Specifies the maximum number of entries to store in the Entry Owner cache. This cache holds information regarding ACL definitions within the database. Retrieval of information from this cache avoids database read operations when resolving access permissions.

The maximum size of this value is 2147483647. A value of 0 indicates that the cache is not used.
Default = 100

extendedGroupSearching {on | off}

Global	TDBM	SDBM	GDBM	EXOP
	X			

Specifies whether a backend participates in extended group membership searching on a client bind request. If this option is **on**, group memberships are gathered from this backend during LDAP bind processing in addition to the backend in which the bind DN resides. If this option is **off**, group memberships are not gathered from this backend unless the bind DN resides in this backend.

The group memberships gathered on a client's LDAP bind request are used for authorization checking of the client's request. The administrator should know, in general, which backends may contain group information so they can be marked for **extendedGroupSearching**. Group memberships are necessary for complete authorization checking of a client request. The setting of the **extendedGroupSearching** configuration option allows the backend to search for static and nested groups which the bind DN may belong to if the bind DN does not exist as an actual entry in any of the TDBM backends on the LDAP server.

The server control **authenticateOnly** is supported by the LDAP server so that a client can override both **extendedGroupSearching** and group membership gathering from the backend where the DN resides. See Appendix D, “Supported server controls,” on page 459 for more information.

This option applies only to the backend in which it is defined.
Default = **off**

filterCacheBypassLimit *num-entries*

Global	TDBM	SDBM	GDBM	EXOP
	X		X	

Specifies the maximum number of returned entries allowed in the result set of any individual search that will be stored in the Search Filter cache. Search filters that match more than this number of entries will not be added to the Search Filter cache. This option is useful for maintaining the effectiveness of the Search Filter cache and Entry cache. It can be used to prevent a few search requests with large result sets from dominating the contents of the Entry cache.

This parameter is ignored when the filter cache is not in use.
The minimum size of this value is 1.
The maximum size of this value is 250.

Default = 100.

filterCacheSize *num-filters*

Global	TDBM	SDBM	GDBM	EXOP
	X		X	

Specifies the maximum number of filters to store in the Search Filter cache. This cache holds information related to the mapping of search request inputs and the result set. Retrieval of information from this cache avoids database read operations when processing search requests. Individual search requests which return more entries than specified in the filterCacheBypassLimit setting are not placed in the cache.

If this option is specified in multi-server mode, a warning message is issued and a value of 0 is used.

The maximum size of this value is 2147483647. A value of 0 indicates that the cache is not used.
TDBM Default = 5000 in single-server mode, 0 in multi-server mode.
GDBM Default = 0.

idleConnectionTimeout *num-seconds*

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies the amount of time in seconds that the LDAP server will wait on an idle connection before timing out on a particular client connection.
Default = 0 (indefinitely)
Recommended value = 1800 (30 minutes)
Minimum value = 30 seconds

Note: The setting of the **idleConnectionTimeout** configuration option is strongly discouraged. When **idleConnectionTimeout** is set, large memory blocks are allocated for each client operation that is being processed by the LDAP server. These memory blocks are persistent until the **idleConnectionTimeout** value is reached or the client application unbinds from

the LDAP server. Storage abends of the LDAP server can occur if there are many of these large memory blocks. Thus, it is not recommended to set the **idleConnectionTimeout** configuration option unless directed to by IBM support.

include *filename*

Global	TDBM	SDBM	GDBM	EXOP
X	X	X	X	X

Specifies the path and file name of a file to be included as a part of the LDAP server configuration.

Refer to page 56 for information on specifying *filename*.

Note that the LDAP server will not detect loop conditions in a set of included files. Configuration may encounter errors or fail if the same file is processed more than once. While nested include files are supported, including the same file in such a way as to form a loop condition is not supported.

index *attrlist* **[eq | none]**

Global	TDBM	SDBM	GDBM	EXOP

Note: The **index** option is now ignored by the LDAP server.

krbIdentityMap [on | off]

Global	TDBM	SDBM	GDBM	EXOP
	X	X		

Specifies if this backend will participate in Kerberos identity mapping. If it will participate, then the server will attempt to map the Kerberos identity that performed the bind to DNs that exist in this backend. The mapped DNs will then be used for access control.

Default = **off**

krbKeytab {*serverKeytab* **| none}**

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies the keytab that will be used for the LDAP server. Following is an example:

```
krbKeytab /home/users/u1/keytab
```

The value for this option must be specified if the KDC does not exist on the same machine as the LDAP server. However, if the KDC resides on the same machine as the LDAP server and the user ID the LDAP server runs under has permission to read from the IRR.RUSERMAP facility class in RACF, then the **krbKeytab** value should be set to **none**.

krbLDAPAdmin *kerberosIdentityDN*

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies the Kerberos identity that represents the LDAP administrator. This option allows the administrator to bind through Kerberos and still maintain administrative authority. The value for this option must be specified as a DN with the attribute type of **ibm-kn**. Following is an example:

```
krbLDAPAdmin ibm-kn=LDAPAdmin@myrealm.com
```

listen *ldap_URL*

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies, in LDAP URL format, the IP address (or host name) and the port number where the LDAP server will listen to incoming client requests. This parameter may be specified more than once in the configuration file.

Note that the **listen** value may be established in the configuration file, or it may be established using the optional start-up parameter for **listen** (see "Setting up and running the LDAP server as a started task" on page 101).

Default = **INADDR_ANY** (that is, ldap://:389) on the nonsecure IPv4 default port of 389.

The format of *ldap_URL* for the **listen** option to listen on a TCP/IP socket interface is the following:

{**ldap://** | **ldaps://**}[*IP_address* | *hostname*][**:***portNumber*]

The format of *ldap_URL* for the **listen** option to listen on the z/OS SAF interface is the following:

{**ldap://** | **ldaps://**}**:pc**

where:

ldap:// Specifies that the server listen on nonsecure addresses or ports. Note that if SSL/TLS is configured for the server, then once a connection is established, the client may switch to secure communication using the Start TLS Extended Operation.

ldaps://
Used to have the server listen on secure addresses or ports. Once a connection is established to the server, the client must begin the SSL/TLS handshake protocol.

IP_address
Specifies either the IPv4 or IPv6 address.

hostname
Specifies the host name. If the host name is used for the **listen** option, all of the IPv4 or IPv6 addresses associated with the *hostname* are obtained from the DNS (Domain Name Server) and the LDAP server listens on each of these IP addresses.

portNumber
Specifies the port number. The *portNumber* is optional. If the port number is not specified for an **ldap://**, then the default of 389 is used for nonsecure connections. If the port number is not specified for an **ldaps://**, then the default of 636 is used for secure connections.

Range = 1 - 65536

If the **sysplexGroupName** and **sysplexServerName** options are present in the configuration file, the port number specified for this server instance must be the same as the port number specified for all other members of the same *group_name* in the sysplex for dynamic workload balancing to function properly.

It is advisable to reserve the port number or numbers chosen here in your TCP/IP profile data set. Also, be aware that port numbers below 1024 may require additional specifications. Consult *z/OS Communications Server: IP Configuration Reference* for more information.

pc Specifies that the LDAP server should listen for program call (PC) calls from Policy Director or RACF change logging using the z/OS Security Authorization Facility (SAF) interface.

Note that when the **listen** option is initialized to listen for PC calls on the LDAP server, the **listen** parameter must not include an IP address or a host name.

Also, there is no difference if you specify **ldap** or **ldaps** as part of *ldap_URL*. Both produce the same result.

Following are some examples of how you can specify *ldap_URL*.

- If you specify:

  ```
  ldap://
  ```

  the LDAP server binds and listens on all available IPv4 addresses (**INADDR_ANY**) on the system on the nonsecure default port of 389.
- If you specify:

  ```
  ldap://us.endicott.ibm.com:489
  ```

  the LDAP server binds and listens on all of the IPv4 and IPv6 addresses associated with host name `us.endicott.ibm.com` on the nonsecure port of `489` for incoming client requests.
- If you specify:

  ```
  ldap://9.130.77.27
  ```

  the LDAP server binds and listens on IPv4 address `9.130.77.27` on the default nonsecure port of `389` for incoming client requests.
- If you specify:

  ```
  ldaps://us.endicott.ibm.com
  ```

  the LDAP server binds and listens for incoming client requests on the IPv4 and IPv6 addresses associated with host name `us.endicott.ibm.com` on the default secure port of `636`.
- If you specify:

  ```
  ldaps://9.130.77.27:736
  ```

  the LDAP server binds and listens on IPv4 address `9.130.77.27` on the secure port of `736`.
- If you specify:

  ```
  ldap://:489
  ```

  the LDAP server binds and listens on all available IPv4 addresses (**INADDR_ANY**) on the system on the nonsecure port of `489` for incoming client requests
- If you specify:

  ```
  ldaps://:777
  ```

  the LDAP server binds and listens on all available IPv4 addresses (**INADDR_ANY**) on the system on the secure port of `777` for incoming client requests.
- If you specify:

  ```
  ldap://[5f1b:df00:ce3e:e200:20:800:2078:e3e3]:389
  ```

  the LDAP server binds and listens on the IPv6 address `5f1b:df00:ce3e:e200:20:800:2078:e3e3` on the system on the nonsecure port of `389` for incoming client requests.
- If you specify:

  ```
  ldaps://[::ffff:9.130.77.75]:777
  ```

the LDAP server binds and listens on the IPv4 mapped IPv6 address of `::ffff:9.130.77.75` on the system on the secure port of `777` for incoming client requests.

- If you specify:

  ```
  ldap://[::]
  ```

  the LDAP server binds and listens on all available IPv4 (**INADDR_ANY**) and IPv6 (**in6addr_any**) addresses on the nonsecure default port of 389.

- If you specify:

  ```
  ldap://:pc
  ```

  the LDAP server binds and listens for PC calls from Policy Director and from RACF change logging using the SAF interface into the server.

Note: The **listen** parameter deprecates the **security**, **port**, and **securePort** options in the configuration file. If there is a **listen** option specified in the configuration file along with either **security**, **port**, or **securePort**, the **listen** option takes precedence over what has been specified for **security**, **port**, or **securePort**. If using an earlier version of the configuration file with **security**, **port**, or **securePort**, the LDAP server will be configured to listen on the port numbers specified for **securePort**, **port**, or both depending upon the **security** setting. However, it is highly recommended that the LDAP server be configured using the **listen** option.

Note: When using the **ldapcnf** utility method to configure the LDAP Server, you do not directly specify a **listen** statement. Instead, the **ldapcnf** utility uses the values specified for the PORT and SECUREPORT variables in the **ldap.slapd.profile** input file to form appropriate **listen** statements that it places in the **slapdcnf** output member.

When configuring the LDAP server without using the **ldapcnf** utility, add the appropriate **listen** statements directly to the configuration file (or specify them on the server start command).

logfile *filename*

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies where to place the activity log records when activity logging is enabled. See “Activity Logging” on page 113 for more information.

Refer to page “Specifying a value for a distinguished name” on page 56 for information on specifying the *filename*.

masterServer *ldap_URL*

Global	TDBM	SDBM	GDBM	EXOP
	X			

Specifies the location of this replica’s master server in LDAP URL format.

In the following example, `myldap.server.com` is the host name and `3389` is the port number of the LDAP URL:

```
masterServer ldap://myldap.server.com:3389
```

In the following example, the IPv6 address of `5f1b:df00:ce3e:e200:20:800:2078:e3e3` is the IP address and `389` is the port number of the LDAP URL.

```
masterServer ldap://[5f1b:df00:ce3e:e200:20:800:2078:e3e3]:389
```

masterServerDN *dn*

Global	TDBM	SDBM	GDBM	EXOP
	X			

Specifies the DN allowed to make changes to the read-only replica. The format of the name must be in DN format which is described in Chapter 13, “Data model,” on page 161. The presence of this option indicates that the server instance using this configuration file is a slave. It is recommended, though not necessary, that the DN have the same suffix as one of the **suffix** option values in the configuration file.

“Establishing the administrator DN and the replica server DN and passwords” on page 87 describes how to set up your master server DN.

In order to enter characters in this distinguished name which are outside of the 7-bit ASCII range when expressed in Unicode (or UTF-8), then you must escape these characters. See the description under the **adminDN** configuration option (page 56) for details on how to do this.

masterServerPW *string*

Global	TDBM	SDBM	GDBM	EXOP
	X			

Specifies the password for the **masterServerDN** that will be allowed to make updates. This option is only applicable for a slave LDAP server. See “Establishing the administrator DN and the replica server DN and passwords” on page 87 for additional information about the master server password.

Note: Use of the **masterServerPW** configuration option is strongly discouraged in production environments. Instead, specify your **masterServerDN** as the distinguished name of an existing entry in the directory information tree, including a userpassword. This will eliminate passwords from the configuration file.

maxConnections *num-connections*

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies the maximum number of concurrently connected clients that the LDAP server allows. The number is specified as a positive integer.

Range = 30 to 65535
Default = operating system maximum

The LDAP server limits the number of client connections by restricting the number of file and socket descriptors used by the LDAP server. Some of the descriptors are used by the LDAP server for its own file descriptors and passive socket descriptors. The value specified for this option should take into account that the server uses approximately 10 descriptors for internal functions and will use more depending upon the number of additional sockets used as passive sockets for connection attempts by clients.

The maximum number of client connections is further restricted by:

- The maximum number of files a single process can have concurrently active.

  The MAXFILEPROC statement for BPXPRMxx and the **maxfileproc** option on the RACF **altuser** command are used to set the limit. Only processes with superuser authority can adjust the limit beyond the limit specified by MAXFILEPROC. Attempts to exceed this limit by non-superuser processes may be audited by the security manager.
- The maximum number of sockets allowed by the TCP/IP socket file system.

The MAXSOCKETS option on the NETWORK statement for BPXPRMxx sets this limit.

Setting these limits too high can affect system performance by using too many resources and deprive other functions of their share of the same resources.

maxThreads *num-threads*

Global	TDBM	SDBM	GDBM	EXOP
X				

Note: The **maxThreads** option is now ignored by the LDAP server. Use the **commThreads** option to specify the number of threads that will be created to handle incoming work from the clients. Refer to page 59 for a description of the **commThreads** option.

multiserver {Y | y | N | n}

Global	TDBM	SDBM	GDBM	EXOP
	X		X	

Indicates the operating mode in which this server will run. Specifying either **y** or **Y** indicates the server runs in multi-server mode with or without dynamic workload management enabled (see page 82 for a description of multi-server operating modes). Specifying either **n** or **N** indicates the server runs in single-server mode.

If **n** or **N** is specified, and both **sysplexGroupName** and **sysplexServerName** options are present in the configuration file, the **multiserver** option value is overridden and the server operates in multi-server mode.

The **multiserver** keyword may be present without the **sysplexGroupName** and **sysplexServerName** keywords, in which case sysplex Workload Management features are disabled.

If **sysplexGroupName**, **sysplexServerName**, and **multiserver** keywords are all omitted from the server configuration file, the server will operate in single-server mode and replication will be enabled. When the server is operating in multi-server mode, replication is disabled in the server.

nativeAuthSubtree {all | *dn*}

Global	TDBM	SDBM	GDBM	EXOP
	X			

Specifies the distinguished name of a subtree where all of its entries are eligible to participate in native authentication. This parameter can appear zero or more times to specify all subtrees that use native authentication. If this parameter is omitted, contains no value, or is set to **all**, then the entire directory is subject to native authentication. This option is ignored if **useNativeAuth selected** or **all** is not specified.

nativeUpdateAllowed {on | yes | off | no}

Global	TDBM	SDBM	GDBM	EXOP
	X			

Enables native password changes in the Security Server to occur through the TDBM backend if the **useNativeAuth selected** or **all** option is specified.

Default = **off**

objectclass *name* **[requires** *attrs***] [allow** *attrs***]**

Global	TDBM	SDBM	GDBM	EXOP
X				

Note: The **objectclass** option is now ignored by the LDAP server.

peerServerDN *dn*

Global	TDBM	SDBM	GDBM	EXOP
	X			

Specifies the DN allowed to make changes to this peer replica. The format of the name must be in DN format which is described in Chapter 13, “Data model,” on page 161. The presence of this option indicates that the server instance using this configuration file is a peer replica. It is recommended, though not necessary, that the DN have the same suffix as one of the **suffix** option values in the configuration file.

“Establishing the administrator DN and the replica server DN and passwords” on page 87 describes how to set up your peer replica DN.

In order to enter characters in this distinguished name which are outside of the 7-bit ASCII range when expressed in Unicode (or UTF-8), then you must escape these characters. See the description under the **adminDN** configuration option (page 56) for details on how to do this.

peerServerPW *string*

Global	TDBM	SDBM	GDBM	EXOP
	X			

Specifies the password for the **peerServerDN** that will be allowed to make updates. This option is only applicable for a peer replica LDAP server. See “Establishing the administrator DN and the replica server DN and passwords” on page 87 for additional information about the peer server password.

Note: Use of the **peerServerPW** configuration option is strongly discouraged in production environments. Instead, specify your **peerServerDN** as the distinguished name of an existing entry in the directory information tree, including a userpassword. This will eliminate passwords from the configuration file.

persistentSearch {yes | no}

Global	TDBM	SDBM	GDBM	EXOP
	X		X	

Allows or disallows persistent search for changes made to entries in a backend. When **no** is specified, persistent search requests for this backend are rejected. See “PersistentSearch” on page 461 for more information on persistent search.

Default = **no**

pcThreads *num-threads*

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies the number of threads to be initialized to handle incoming program call (PC) calls using the z/OS SAF interface into the LDAP server.

Default = 10

port *num-port*

Global	TDBM	SDBM	GDBM	EXOP
X				

Note: The **port** option is deprecated when the **listen** option is specified. See page 65 for information on the **listen** option.

TCP/IP port used for non-SSL communications.
Range = 1 - 65535
Default = 389

If the **sysplexGroupName** and **sysplexServerName** options are present in the configuration file, the port number specified for this server instance must be the same as the port number specified for all other members of the same *group_name* in the sysplex for dynamic workload balancing to function properly. Note that the port number may be established in the configuration file, or it may be established using the optional startup parameter for port (see "Setting up and running the LDAP server in the z/OS shell" on page 105).

It is advisable to reserve the port number chosen here in your TCP/IP profile data set. Consult *z/OS Communications Server: IP Configuration Guide* for further information.

pwEncryption {none | crypt | MD5 | SHA | DES:*keylabel***}**

Global	TDBM	SDBM	GDBM	EXOP
	X			

Specifies what encryption method to use when storing the **userPassword** attribute values in the backend of the directory.

none Specifies no encryption. Specifies the **userPassword** attribute values are stored in clear text format. Note that these clear text passwords are stored in the directory prefixed with the tag {none}.

crypt Specifies that **userPassword** attribute values are encoded by the crypt() algorithm before they are stored in the directory. These passwords are stored in the directory prefixed with the tag {crypt}.

MD5 Specifies that **userPassword** attribute values are encoded by the MD5 hash algorithm using OCSF before they are stored in the directory. These passwords are stored in the directory prefixed with the tag {MD5}.

SHA Specifies that **userPassword** attribute values are encoded by the SHA hash algorithm using OCSF before they are stored in the directory. These passwords are stored in the directory prefixed with the tag {SHA}.

DES:*keylabel*
Specifies that **userPassword** attribute values are encrypted by the DES algorithm using the specified key label using OCSF and ICSF before they are stored in the directory, and can be retrieved as part of an entry in the original clear text format. These passwords stored in the directory are prefixed with the tag and key label {DES:*keylabel*}. Retrieval will continue to be limited by the access controls in effect on the entry that contains the **userPassword** attribute values.

The *keylabel* must refer to a valid single-length data-encrypting key, also called a data key, generated by KGUP of ICSF and stored in the CKDS. The *keylabel* maximum length is 64 characters. See the information on managing cryptographic keys and using the Key Generator Utility Program in *z/OS Cryptographic Services ICSF Administrator's Guide* for

instructions on how to generate, store into CKDS, and authorize a data key and key label for DES encryption. It is important to remember to refresh both CKDS and RACF after you enter and authorize a key.

Notes:

1. When an encrypted password is stored in the TDBM backend, it is prefixed with the appropriate encryption tag so that when a clear text password is sent on an LDAP API simple bind it can be encrypted in that same method for password verification.
2. The crypt() algorithm, implemented across many platforms, accepts only the first eight characters of a password. As a result, any password supplied on an **ldap_simple_bind** operation or **ldap_compare** operation that matches the first eight characters of a **userPassword** attribute value encrypted with the crypt() algorithm in the directory will match.
3. The values returned by the crypt() algorithm are not portable to other X/Open-conformant systems. This means that user password values encoded by the crypt() algorithm and unloaded as tagged output using **tdbm2ldif -t** are not portable when loaded by another platform's load utility.

If **pwEncryption** is not specified in the configuration file, no password encryption takes place. User passwords are stored in clear text format and no tag is prefixed.

readOnly {on | off}

Global	TDBM	SDBM	GDBM	EXOP
	X	X	X	

Specifies the ability to modify the database. Any attempt to use the LDAP server to modify the database will fail if **readOnly** is turned **on**.

Note: For GDBM, change log entries continue to be created and trimmed (deleted) by the LDAP server even when **readOnly** is on.

Default = **off**

referral *ldap_URL*

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies the referral to pass back when the local database cannot handle the request. It is also known as the default referral. The **referral** option can appear multiple times and should list equivalent servers.

In the following example, `myldap.server.com` is the host name and `3389` is the port number of the LDAP URL:

```
referral ldap://myldap.server.com:3389
```

In the following example, the IPv6 address `5f1b:df00:ce3e:e200:20:800:2078:e3e3` is the IP address and `389` is the port number of the LDAP URL:

```
referral ldap://[5f1b:df00:ce3e:e200:20:800:2078:e3e3]:389
```

schemaReplaceByValue {on | off}

Global	TDBM	SDBM	GDBM	EXOP
	X		X	

Determines the behavior of modify operations with replace values of the schema entry. When **schemaReplaceByValue off** is specified, a modify operation with replace values for an attribute in

the schema entry behaves like a 'normal' modify operation: all the values currently in the attribute are replaced by the values specified in the modify operation. When **schemaReplaceByValue on** is specified, individual values in an attribute in the schema entry can be replaced without removing all the other values currently in the attribute. See "Updating the schema" on page 179 for more information on modifying the schema.

The **schemaReplaceByValue** configuration option can be overridden on a specific modify operation by including the **schemaReplaceByValueControl** control in the modify request.
Default = **on**

securePort *num-port*

Global	TDBM	SDBM	GDBM	EXOP
X				

Note: The **securePort** option has been deprecated by the **listen** option. See page 65 for information on the **listen** option.

TCP/IP port used for SSL communications.
Range = 1 - 65535
Default = 636

It is advisable to reserve the port number chosen here in your TCP/IP profile data set. Consult *z/OS Communications Server: IP Configuration Guide* for further information.

security {ssl | sslonly | none | nossl}

Global	TDBM	SDBM	GDBM	EXOP
X				

Note: The **security** option has been deprecated by the **listen** option. See page 65 for information on the **listen** option.

Specifies what type of communications will be accepted. The **ssl** setting indicates that the server will listen on the secure port as well as the non-secure port. The **sslonly** setting means that the server will listen only on the secure port. The **none** or **nossl** settings indicate that the server will listen only on the non-secure port.
Default = **none**

sendV3stringsoverV2as {UTF-8 | ISO8859-1}

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies the output data format to use when sending **UTF-8** information over the LDAP Version 2 protocol.
Default = **UTF-8**

See "UTF-8 data over the LDAP Version 2 protocol" on page 336 for more detailed information on the use of this setting.

serverEtherAddr *mac_address*

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies the Media Access Control (MAC) address used for entry UUID generation. This value must be unique for all LDAP servers in your enterprise. You must specify the MAC address if multiple LDAP servers will run on a (hardware) system. This applies if your LDAP servers are on different LPARs and also if two LDAP servers are on the same LPAR. You do not need to specify this field if this is the only LDAP server that will run on this (hardware) system.

The suggested form of the *mac_address* you place here is:

```
4xmmmmssssss
```

Where:

x Is a one-character LDAP number. If more than one LDAP server is operating on a CPU, specify a different *x* value for each server. If more than 16 LDAP servers are desired, then use a serial number and model number from a CPU that is not running an LDAP server. If another CPU is not available, then set the *x*, *mmmm*, and *ssssss* values from the MAC address on an old Ethernet card that is no longer being used or not used to run an LDAP server.

mmmm Is the four-digit model number of the CPU.

ssssss Is the six-digit serial number of the CPU.

Note that the allowable values for *x*, *mmmm*, and *ssssss* are: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

Following is an example using the **serverEtherAddr** option:

```
serverEtherAddr 4A123401234D
```

serverKrbPrinc *kerberosIdentity*

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies the server's Kerberos identity that was created in "Defining the Kerberos identity" on page 38. This value is used to acquire the server credentials. The format for *kerberosIdentity* is:

```
ldap_prefix/hostname@realm-name
```

where *ldap_prefix* is **ldap** or **LDAP**

hostname is the primary host name in DNS of the system on which the LDAP server is running

realm-name is the Kerberos realm name specified in the Kerberos configuration file.

The following are examples of **serverKrbPrinc**:

```
serverKrbPrinc LDAP/myhost.myrealm.com@myrealm.com
serverKrbPrinc ldap/myhost.newrealm.com@newrealm.com
```

servername *string*

Global	TDBM	SDBM	GDBM	EXOP
	X		X	

Specifies the name of the DB2 server location that manages the tables for the LDAP Server. This value must match the name of one of the `DATA SOURCE` stanzas that must be specified in the ODBC initialization data set which is specified by the **dsnaoini** option in the configuration file. See *DB2 ODBC Guide and Reference* for a description of the DSNAOINI ODBC initialization data set contents. Using the example DSNAOINI file in Figure 2 on page 15 the value of *string* for **servername** would be `LOC1`.

sizeLimit *num-limit*

Global	TDBM	SDBM	GDBM	EXOP
X	X	X	X	

Specifies the maximum number of entries to return from a search operation. The maximum number can be modified on a specific search request as described below.
Range = 0 - 2147483647
0 = no limit
Default = 500

This option applies to all backends, except EXOP, unless specifically overridden in a backend definition. Specifying this prior to a **database** line in the configuration sets the option for all backends, except EXOP. Specifying it after a **database** line sets the option just for the backend defined by the **database** line.

A limit on the number of entries returned can also be specified by the client on a search request. Note that the following behavior is used when referring to the **sizeLimit** parameter.

When accessing the z/OS LDAP server using the TDBM or GDBM backend:

- If the client has not bound as the **adminDN**, then the limit is the smaller of the limit passed by the client and the limit read by the server from the **sizeLimit** record in **slapd.conf** (which defaults to 500). If the client does not specify a limit, then the server limit is used.
- If the client has bound as the **adminDN**, then the limit is the value passed by the client. If the client does not specify a limit, then the number of entries returned is unlimited. The limit from the **slapd.conf** file is ignored when the client has bound as the **adminDN**.

When accessing the z/OS LDAP support for RACF (the SDBM backend):

- The limit is the smaller of the limit passed by the client and the limit read by the server from the **sizeLimit** record in **slapd.conf** (which defaults to 500). If the client does not specify a limit, then the server limit is used. It does not matter how the client has bound.
- The number of entries returned may be further restricted by limits imposed by RACF. See the information about accessing RACF information in Chapter 16, “Accessing RACF information,” on page 213.

sslAuth {serverAuth | serverClientAuth}

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies the SSL/TLS authentication method. The **serverAuth** method allows the LDAP client to validate the LDAP server on the initial contact between the client and the server. The **serverAuth** method is the default.

The **serverClientAuth** method allows the LDAP client to validate the LDAP server and the LDAP server to validate the LDAP client if the client sends its digital certificate on the initial contact between the client and the server.

Note: To allow clients to **SASL EXTERNAL** bind to the LDAP server, it is necessary to configure the server with **sslAuth serverClientAuth**.

See “Setting up for SSL/TLS” on page 46 for more SSL/TLS information.

sslCertificate {*certificateLabel* **| none}**

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies the label of the certificate that will be used for server authentication. The certificate is stored in the key database file which is created and managed using the **gskkyman** tool. See *z/OS Crytographic Services System Secure Sockets Layer Programming* for details on using the **gskkyman** tool.

Default = **none**

If the value is **none** (by default or by specification), the default certificate, marked in the key database file managed by the **gskkyman** tool, will be used for server authentication.

sslCipherSpecs {*string* **| ANY}**

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies the SSL/TLS cipher specifications that will be accepted from clients. The cipher specification is a blank delimitted string that represents an ORed bitmask indicating the SSL/TLS cipher specifications that will be accepted from clients. Clients that support any of the specified cipher specifications will be able to establish an SSL/TLS connection with the server. Table 11 shows a list of the cipher spec mask values and the related decimal, hexidecimal, and keyword values. Refer to *z/OS Crytographic Services System Secure Sockets Layer Programming* for a description of supported cipher specifications.

The cipher specification may be specified as follows:

- A decimal value (for example, 256)
- A hexadecimal value (for example, x100)
- A keyword (for example, **TRIPLE_DES_SHA_US**)
- A construct of those values using plus and minus signs to indicate inclusion or exclusion of a value. For example,
  - `256+512` is the same as specifying `768`, or `x100+x200`, or `TRIPLE_DES_SHA_US+DES_SHA_US`
  - `52992` is the same as specifying `ALL-RC2_MD5_EXPORT-RC4_MD5_EXPORT`

Table 11. Supported ciphers

Cipher	Hexadecimal value	Decimal value
TRIPLE_DES_SHA_US	x0100	256
DES_SHA_EXPORT	x0200	512
RC4_SHA_US	x0400	1024
RC4_MD5_US	x0800	2048
RC2_MD5_EXPORT	x1000	4096
RC4_MD5_EXPORT	x2000	8192
RSA_AES_128_SHA	x4000	16384
RSA_AES_256_SHA	x8000	32768
ANY	xFF00	65280
ALL	xFF00	65280

Depending upon the level of System SSL support installed, some ciphers may not be supported. System SSL will ignore the unsupported ciphers. You should consult the System SSL documentation to determine the specific ciphers that your installation supports.

See “Setting up for SSL/TLS” on page 46 for more SSL/TLS information.

Default = **ANY**

sslKeyRingFile *name*

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies either the path and file name of the SSL/TLS key database file for the server or the name of the RACF key ring for the server.

The file name must match the key database file name that was created using the **gskkyman** utility (see *z/OS Crytographic Services System Secure Sockets Layer Programming*). Also, see "Setting up for SSL/TLS" on page 46 for more SSL information.

The LDAP server supports the use of a RACF key ring. Specify the RACF key ring name for the **sslKeyRingFile** and comment out the **sslKeyRingFilePW** and **sslKeyRingPWStashFile** options to use this support. Also, see "Creating and using a key database or key ring" on page 48 for more information on using RACF key rings.

sslKeyRingFilePW *string*

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies the password protecting access to the SSL/TLS key database file. The password string must match the password to the key database file that was created using the **gskkyman** utility (see *z/OS Crytographic Services System Secure Sockets Layer Programming*). Also, see "Setting up for SSL/TLS" on page 46 for more SSL information.

Note: Use of the **sslKeyRingFilePW** configuration option is strongly discouraged. As an alternative, use either the RACF key ring support or the **sslKeyRingPWStashFile** configuration option. This will eliminate this password from the configuration file.

Comment out the **sslKeyRingFilePW** and **sslKeyRingPWStashFile** options if you are using RACF for the key ring.

sslKeyRingPWStashFile *filename*

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies an HFS file name where the password for the server's key database file is stashed. Use the full path name of the stash file in HFS for *filename*.

If this option is present, then the password from this stash file overrides the **sslKeyRingFilePW** configuration option, if present. Use the **gskkyman** utility with the **-s** option to create a key database password stash file. See "Setting up for SSL/TLS" on page 46 for more SSL information.

Comment out the **sslKeyRingFilePW** and **sslKeyRingPWStashFile** options if you are using RACF for the key ring.

suffix *dn_suffix*

Global	TDBM	SDBM	GDBM	EXOP
	X	X		

Denotes the root of a subtree in the namespace managed by this server within this backend. This option may be specified more than once to indicate all the roots of the subtrees within this backend.

Do not specify identical suffixes. LDAP requests using that suffix will fail because the server does not know to which suffix the request should be directed. Also, avoid using overlapping suffixes, where one suffix is an ancestor of another suffix. These suffixes create confusion and can result in unexpected results. An example of overlapping suffixes is:

```
suffix ou=Server Group, o=IBM
suffix o=IBM
```

If the suffix contains a special character (according to RFC 2253), it must be prefixed by two backslashes (\\). Note that the double backslashes are only needed in the configuration file; in all other usages, the special character is usually prefixed by a single backslash. For example, to use the suffix `o=MyCompany#1`, you must specify `o=MyCompany\\#1` in the configuration file and `o=MyCompany\#1` in other places where the suffix is used (for example, in LDAP search requests). See Chapter 16, "Accessing RACF information," on page 213 for exceptions to this when using SDBM. See IETF RFC 2253 *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names* for valid DN formats.

Domain Component naming as specified by RFC 2247 is also supported in the LDAP server. For example, the domain name `ibm.com` could be specified as the following suffix in the LDAP configuration file:

```
suffix "dc=ibm,dc=com"
```

This option applies only to the backend in which it is defined. The EXOP backend does not require a **suffix** option in the configuration file. If one is specified, it will be ignored. Also, a **suffix** option cannot be specified for the GDBM backend.

When using SDBM, do not specify more than one suffix. An SDBM suffix should not contain an alias name for an attribute. For example, an SDBM suffix cannot use the **surName** attribute (it can use the **sn** attribute instead). Also, you can use a case-sensitive attribute in the suffix, but SDBM ignores case when processing the suffix.

supportKrb5 {yes | no}

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies if the LDAP server participates in Kerberos GSS API Authentication. If it participates, then Kerberos GSS API binds are accepted and information is stored in the server's root DSE.
Default = **no**

sysplexGroupName *group_name*

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies the name of the application group within which this server instance becomes a member for purposes of dynamic workload balancing. All concurrently-running LDAP servers which participate in dynamic workload balancing within the same Parallel Sysplex® (see page 83 for the description of a Parallel Sysplex) must use the same *group_name* in their server configuration file. This name may be up to 18 characters in length, and must be unique among all application groups using Workload Manager services.

The **sysplexGroupName** and **sysplexServerName** keywords are corequisites, and Sysplex Workload Management features will only be enabled if both are present with non-null arguments. When Sysplex Workload Management features are enabled, the server is automatically assumed to be in multi-server mode, and replication will be disabled in this server.

If **sysplexGroupName**, **sysplexServerName**, and **multiserver** keywords are all omitted from the server configuration file, the server will operate in single-server mode and replication will be possible.

sysplexServerName *server_name*

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies the name of the server within the *group_name* sysplex group which participates in dynamic workload balancing. (See page 83 for the description of a sysplex.) All concurrently-running LDAP servers which participate in dynamic workload balancing within the same Parallel Sysplex are identified by *server_name*s within a given *group_name* used in the sysplex. This name may be up to 8 characters in length, and must be unique within a given *group_name* using Workload Manager services.

The **sysplexGroupName** and **sysplexServerName** keywords are corequisites, and Sysplex Workload Management features will only be enabled if both are present with non-null arguments. When Sysplex Workload Management features are enabled, the server is automatically assumed to be in multi-server mode, and replication will be disabled in this server.

If **sysplexGroupName**, **sysplexServerName**, and **multiserver** keywords are all omitted from the server configuration file, the server will operate in single-server mode and replication will be possible.

timeLimit *num-seconds*

Global	TDBM	SDBM	GDBM	EXOP
X	X	X	X	

Specifies the maximum number of seconds (in real time) the LDAP server will spend answering a search request or a Modify DN request. This maximum number can be modified on a specific search request as described below. If a request cannot be processed within this time, a result indicating an exceeded time limit is returned.

Range = 0 - 2147483647
0 = no limit
Default = 3600

This option applies to all backends, except EXOP, unless specifically overridden in a backend definition. Specifying this prior to a **database** line in the configuration sets the option for all backends, except EXOP. Specifying it after a **database** line sets the option just for the backend defined by the **database** line.

A limit on the amount of time can also be specified by the client on a search request. Note that the following behavior is used on a search operation when referring to the **timeLimit** parameter.

When accessing the z/OS LDAP server using the TDBM or GDBM backend:

- If the client has not bound as the **adminDN**, then the limit is the smaller of the limit passed by the client and the limit read by the server from the **timeLimit** record in **slapd.conf** (which defaults to 3600). If the client does not specify a limit, then the server limit is used.
- If the client has bound as the **adminDN**, then the limit is the value passed by the client. If the client does not specify a limit, then there is no time limit. The limit from the **slapd.conf** file is ignored when the client has bound as the **adminDN**.

When accessing the z/OS LDAP support for RACF (the SDBM backend):

- The limit is the smaller of the limit passed by the client and the limit read by the server from the **timeLimit** record in **slapd.conf** (which defaults to 3600). If the client does not specify a limit, then the server limit is used. It does not matter how the client has bound.

useNativeAuth {selected | all | off}

Global	TDBM	SDBM	GDBM	EXOP
	X			

Enables native authentication in the TDBM backend. If the value is:

- **selected**, only entries with the **ibm-nativeId** attribute that are within the native subtrees (see **nativeAuthSubtree** option on page 69) use native authentication.
- **all**, all entries within native subtrees use native authentication. These entries can contain the **ibm-nativeId** or **uid** attribute to specify the RACF ID.
- **off**, no entries participate in native authentication.

Default = **off**

validateincomingV2strings {on | yes | off | no}

Global	TDBM	SDBM	GDBM	EXOP
X				

Specifies whether the incoming strings are validated. If set to **on**, this setting limits the format of incoming string data sent over the LDAP Version 2 protocol to the IA5 character set (X'00'-X'7F' or ″7-bit ASCII″). With this setting, textual data received on operations outside of the IA5 character set causes the operations to fail with **LDAP_PROTOCOL_ERROR**.
Default = **on**

Note that while supported, it is not recommended to run with this data filtering disabled.

verifySchema {on | off}

Global	TDBM	SDBM	GDBM	EXOP
X				

Note: The **verifySchema** option is now ignored by the LDAP server.

waitingThreads *num-threads*

Global	TDBM	SDBM	GDBM	EXOP
X				

Note: The **waitingThreads** option is deprecated and ignored by the LDAP server. The **commThreads** (page 59) and **idleConnectionTimeout** (page 63) options have replaced the **waitingThreads** option.

Deprecated options

The **listen** parameter deprecates the **security**, **port**, and **securePort** options in the configuration file. If a **listen** option is specified in the configuration file with either **security**, **port**, or **securePort**, the **listen** will take precedence over what has been specified for the deprecated **security**, **port**, and **securePort** options. If using an earlier version of the configuration file which contains the **security**, **port**, or **securePort** options, the LDAP server will be configured to listen on the port numbers specified for **securePort**, **port**, or both, depending upon the **security** setting. However, it is highly recommended that the LDAP server be configured using the **listen** option. See the description of the **listen** option on page 65 for more information.

Ignored options

The **replKeyRingFile** and **replKeyRingPW** options are no longer necessary or evaluated by the LDAP server. These options should be removed from the configuration file. Use the **sslKeyRingFile** option to specify the key database file and use the **sslKeyRingPWStashFile** configuration option or the RACF key ring support for the password.

The **maxThreads** and **waitingThreads** options are no longer necessary or evaluated by the LDAP server. These options should also be removed from the configuration file. Use the **commThreads** option to set the number of threads initialized at server start-up for communicating with the clients. See the description of the **commThreads** option on page 59 for more information.

The **attribute**, **objectclass**, and **verifySchema** options are no longer necessary or evaluated by the LDAP server. These options should be removed from the configuration file.

With the removal of the RDBM backend, **rdbm** can no longer be specified as the *dbtype* on the **database** option. The **index**, **tbspaceentry**, **tbspacemutex**, **tbspace32k**, and **tbspace4k** options are no longer necessary or evaluated. These options should be removed from the configuration file.

Configuration considerations

The following table shows all of the different options you have and the decisions you must make for your LDAP server configuration. It also shows where you can find the associated reference information to help you make these decisions.

Table 12. Configuration considerations

Dependency	More information
HFS environment versus PDS (dataset) environment Depending on whether you use an HFS or PDS environment, there are some areas to consider for each.	"Setting up and running the LDAP server in the z/OS shell" on page 105 and "Setting up and running the LDAP server as a started task" on page 101
Operational mode You need to determine the type of operational mode your LDAP server will run in. For example, single-server mode or multi-server mode with or without dynamic workload management enabled.	"Determining operational mode" on page 82
TDBM backend You can use a TDBM backend database based on DB2.	Chapter 14, "LDAP directory schema," on page 165
SDBM backend You can use an SDBM backend database based on RACF.	"Setting up for SDBM" on page 44
GDBM backend You can use a GDBM backend database based on DB2 to log changes to RACF data or TDBM entries.	"Setting up for GDBM" on page 45
EXOP backend You can use an EXOP backend to retrieve Policy Director data.	Chapter 20, "Using extended operations to access Policy Director data," on page 255
SSL/TLS If you want to protect LDAP access with Secure Socket Layer (SSL) or Transport Layer Security (TLS), your LDAP server can be configured to provide server and, optionally, client authentication.	"Setting up for SSL/TLS" on page 46

Table 12. Configuration considerations (continued)

Dependency	More information
Password encryption Your LDAP server can prevent unauthorized access to user passwords in a TDBM backend database.	"Configuring for user password encryption" on page 50
Kerberos authentication You can enable GSS API Kerberos binds and configure identity mapping.	Chapter 17, "Kerberos authentication," on page 233
Native authentication You can enable and configure your directory to perform authentication using the Security Server.	Chapter 18, "Native authentication," on page 241
CRAM-MD5 and DIGEST-MD5 authentication The LDAP server can be configured to perform CRAM-MD5 and DIGEST-MD5 authentication binds.	Chapter 19, "CRAM-MD5 and DIGEST-MD5 Authentication," on page 251
Administrator DN and replica server DN and passwords Determine how to set up your administrator DN and password. Also determine how to set up your master or peer server DN and password, if you are using replication.	"Establishing the administrator DN and the replica server DN and passwords" on page 87
Replication To keep multiple databases in sync, you can use replication.	See Chapter 23, "Replication," on page 289 or "Establishing the administrator DN and the replica server DN and passwords" on page 87 for more information.
Referrals To refer clients to additional directory servers, use referrals.	See Chapter 26, "Referrals," on page 315 for more information.
LDAP entry UUID To generate Media Access Control (MAC) address used for entry UUID	See Chapter 11, "Running and using the LDAP backend utilities," on page 135

"Example configuration scenarios" on page 89 has a variety of examples showing different LDAP server configurations.

Determining operational mode

Note: If you already have the LDAP server installed, see Chapter 10, "Migrating to a z/OS LDAP server," on page 117.

Once the software has been installed, you are ready to configure it for use at your site. The LDAP server may be configured to run in one of several operational modes when a TDBM or GDBM backend is configured.

- **Single-server mode**

  In this operational mode, only a single instance of the LDAP server may use a given TDBM or GDBM database to store directory data. This server may perform replication (see Chapter 23, "Replication," on page 289) of TDBM database changes to other servers (on the same host system or on another host system).

  See "Operating in single-server mode" on page 84 for more information.
- **Multiple single-server mode LDAP servers**

In this operational mode, two or more LDAP servers, each in single-server mode, can be run on the same system with different TDBM or GDBM backends. These servers may perform replication (see Chapter 23, “Replication,” on page 289) of TDBM database changes to other servers (on the same host system or on another host system). However, each server must have its own separate replica.

See “Setting up multiple LDAP servers” on page 87 for more information.

- **Multi-server mode without dynamic workload management enabled**

  In this operational mode, multiple concurrent instances of the LDAP server using the same TDBM or GDBM database to store directory data may run on a given host system, as well as on different host systems when those hosts are coupled in a Parallel Sysplex. A *Parallel Sysplex* is a collection of z/OS systems that cooperate, using certain hardware and software products, to process work. A Parallel Sysplex enables high-performance, multisystem data sharing across multiple Central Processor Complexes and z/OS images, as well as dynamic workload balancing across constituent systems in the sysplex. For additional information, see *z/OS Parallel Sysplex Overview*.

  Multi-server mode is intended for use in an environment where high transactional volume is common, or where maximum availability is required. This mode provides benefits of improved availability, fault tolerance, improved resource utilization, and improved performance. These benefits are achieved by enabling concurrent running of multiple servers which are functionally equivalent and which provide access to the same LDAP data.

  Note: In this operational mode, replication of TDBM database changes to other servers is not supported.

  See “Operating in multi-server mode without dynamic workload management enabled” on page 84 for more information.

- **Multi-server mode with dynamic workload management enabled**

  This operational mode augments the benefits of the previously described mode with dynamic workload management. This mode may only be used when all host systems on which instances of the LDAP server will run are coupled in the same Parallel Sysplex and all instances of the LDAP server are using the same TDBM or GDBM database. The dynamic workload management for LDAP servers is provided through the use of a z/OS TCP/IP feature called ″connection optimization″. Connection optimization uses Domain Name Services (DNS) for distributing connections among server applications within a sysplex domain. Connection optimization achieves workload balancing by distributing connections to systems with the most available resources and by avoiding unavailable sysplex resources. See *z/OS Communications Server: IP Configuration Reference* for information on connection optimization.

  Note: In this operational mode, replication of TDBM database changes to other servers is not supported.

  See “Operating in multi-server mode with dynamic workload management enabled” on page 85 for more information.

- **Program call (PC) callable support mode**

  The program call (PC) callable support in LDAP provides a program call interface to the LDAP extended operations backend (EXOP). This interface is only available using the z/OS SAF interfaces designed to allow Policy Director access to LDAP data and to allow RACF to log changes to RACF data in the LDAP change log.

  See “Operating in PC callable support mode” on page 86 for more information.

In any of these modes, all combinations of TDBM (one or more), SDBM, GDBM, and EXOP backends are supported. The GDBM backend requires the SDBM backend to create change log entries for changes to RACF data.

Notes:

1. A single LDAP server instance can have one SDBM backend and one GDBM backend, but it can have multiple TDBM backend instances

2. If multiple single-server mode LDAP servers are being used on the same system, only one of the LDAP servers can be configured for PC callable support.
3. If multi-server mode is being used and RACF data will be accessed from both servers, then the RACF database should also be shared across the systems where the LDAP servers run to ensure consistency of SDBM operations.

Operating in single-server mode

For the LDAP server to operate in single-server mode, the server configuration file may contain any of the previously documented options except the **sysplexGroupName** and **sysplexServerName** options (the presence of which causes the LDAP server to operate in multi-server mode with dynamic workload management enabled). If the **multiserver** option is present, its value must be set to either **n** or **N**.

Restrictions

If one LDAP server instance using a given DB2-based backend to store directory information is operating in single-server mode, it must be the *only* instance of the LDAP server using that DB2-based backend. Configuring more than one LDAP server instance to use the same DB2 database may yield unpredictable results if one or more of those server instances is configured in single-server mode. If it is desired to access the same DB2-based backend from more than one server instance, all server instances using the same DB2-based backend must be configured to operate in multi-server mode.

Operating in multi-server mode without dynamic workload management enabled

For the LDAP server to operate in multi-server mode without dynamic workload management enabled, the server configuration file may contain any of the previously documented options *except* the **sysplexGroupName** and **sysplexServerName** options. If either of these keywords are present, they cause the LDAP server to operate in multi-server mode with dynamic workload management enabled. The **multiserver** mode option must be present with a value of **y** or **Y**. All instances of the LDAP server using the same DB2-based backend must have the **multiserver** option present in the configuration file used to start each server instance when operating in this mode.

When you are using referrals without dynamic workload management enabled, multiple default referrals defined for other servers can be set up to point to each of the multiple server instances. Similarly, any referral objects defined in other servers which point to the multiple server instances can have multi-valued **ref** attributes set up, each of which is an LDAP URL pointing to the corresponding server instances.

Dependencies

The LDAP server may operate in multi-server mode without dynamic workload management enabled. It is still possible to run multiple concurrent server instances configured to use the same DB2-based backend on the same z/OS image, and/or run multiple concurrent server instances on multiple z/OS images coupled in a Parallel Sysplex. If multiple instances will be run on multiple z/OS images in a Parallel Sysplex, the DB2 subsystems to which each server instance will attach (see “Getting DB2 installed and set up for CLI and ODBC” on page 13) must be configured on each of the images as members of the same DB2 data sharing group. (See *DB2 Data Sharing: Planning and Administration* for information on planning, installing, and enabling DB2 data sharing, and *z/OS MVS Setting Up a Sysplex* for information on planning and installing a Parallel Sysplex.)

Restrictions

If multiple LDAP server instances are using the same DB2-based backend to store directory information, all LDAP server instances using that database must be operating in multi-server mode, either with or without dynamic workload management enabled.

Note: When configuring multiple servers to access the same set of DB2 tables, the **dbuserid** option must be set to the same value for all the servers that are accessing the same tables.

LDAP server instances operating in multi-server mode, either with or without dynamic workload management enabled, will not perform replication (see Chapter 23, “Replication,” on page 289), even if replication objects are present in the DB2-based backend (TDBM). If replication is required, single-server mode must be used.

Operating in multi-server mode with dynamic workload management enabled

For the LDAP server to operate in multi-server mode with dynamic workload management enabled, the server configuration file may contain any of the previously documented options. The **sysplexGroupName** and **sysplexServerName** options must both be present in the server configuration file. If the **multiserver** option is present with a value of **n** or **N**, the option value is overridden and treated as though it were set to **Y**.

To exploit the dynamic workload management feature, the z/OS images on which the LDAP server runs must be coupled in a Parallel Sysplex. Name servers must be configured for connection optimization and started in the same sysplex in which the LDAP server instances are running. See “Dependencies” for more information.

When multiple concurrent instances of the LDAP server are operating in multi-server mode with dynamic workload management enabled, server instances within the same group are considered to provide equivalent service. In essence, the servers are treated as clones of each other. With this in mind, it should be noted that the port on which the server is started should be the same for each instance.

To connect to any unspecified server instance in **sysplexGroupName** *group_name*, the client specifies a target host name of *group_name*.*sysplex_domain_name*, where *group_name* is the name of the application group in which servers are configured using the **sysplexGroupName** option in the server configuration file used to start each respective LDAP server instance, and *sysplex_domain_name* is the name or alias for the sysplex domain. The client will be connected to a server instance in the group on the system in the sysplex with the most available resources, at the port specified by the client which must agree with the port configured when the server instances in the group were started. While it is possible to specify a particular server instance using a fully-qualified server name, doing so reduces the effectiveness of dynamic workload management through connection optimization.

Also note that it is possible to run multiple concurrent LDAP servers using the same DB2-based backend with a mix of servers enabled or not enabled for dynamic workload management, but doing so also reduces the effectiveness of dynamic workload management through connection optimization.

When you are using referrals with dynamic workload management enabled, a single default referral defined at other servers is used where the *host* is specified as *sysplexGroupName*.*sysplexDomainName* and similarly, referral objects defined in other servers use this as the host within a single-valued **ref** attribute.

Dependencies

The LDAP server may operate in multi-server mode with dynamic workload management enabled while running multiple concurrent server instances configured to use the same DB2-based backend on multiple z/OS images coupled in a parallel sysplex. The DB2 subsystems to which each server instance will attach (see “Getting DB2 installed and set up for CLI and ODBC” on page 13) must be configured on each of the images as members of the same DB2 data sharing group. (See *DB2 Data Sharing: Planning and Administration* for information on planning, installing, and enabling DB2 data sharing, and *z/OS MVS Setting Up a Sysplex* for information on planning and installing a Parallel Sysplex.)

Name servers must be configured for connection optimization and started in the same sysplex in which the LDAP server instances are running. Proper distribution of server application addresses for connection

optimization to function properly requires DNS queries to be answered by the name server within the sysplex. For this reason, name servers located outside the sysplex cannot be configured as primary or secondary servers for the sysplex domain.

The port numbers on which the LDAP server instances will listen, both secure and unsecure, must be the same for all servers in the same **sysplexGroupName** for the dynamic workload management to be effective.

Also, the Workload Management Services on each host in the sysplex must be configured in "goal mode". (See *z/OS Communications Server: IP Configuration Reference* for information on configuring a sysplex domain for connection optimization and for information on how to configure Workload Management Services in goal mode.)

Restrictions

If multiple LDAP server instances are using the same DB2-based backend to store directory information, all LDAP server instances using that database must be operating in multi-server mode, either with or without dynamic workload management enabled.

Note: When configuring multiple servers to access the same set of DB2 tables, the **dbuserid** option must be set to the same value for all the servers that are accessing the same tables.

LDAP server instances operating in multi-server mode, either with or without dynamic workload management enabled, will not perform replication (see Chapter 23, "Replication," on page 289), even if replication objects are present in the DB2 database.

Operating in PC callable support mode

The program call (PC) callable support in LDAP provides a program call interface to the LDAP extended operations backend (EXOP) and to the change log backend (GDBM). This interface is only available using the z/OS SAF interfaces designed to allow Policy Director access to LDAP data and to allow RACF to log changes to RACF data in the LDAP change log. The PC callable support is initialized in an LDAP server when the appropriate **listen** option is included in the configuration file or specified when starting the server. An LDAP server can be dedicated to running just the PC callable support or it can run the PC callable support in addition to its normal socket interfaces.

Running the PC callable support has two interactions with the system:

- The address space of the LDAP server is made non-swappable during initialization of the PC callable support. As a result, resources used by that address space can significantly affect system performance.
- Because the PC callable support connects its PC table to a system index, the address space identifier of the LDAP server address space is not re-usable until the next IPL. If the system is configured with a low limit on the number of address spaces, it is possible to run out of address space identifiers, preventing new address spaces from being started. This problem is more likely to occur if the LDAP server running PC callable support is frequently brought down and re-started.

When using the PC callable support for Policy Director access to LDAP data, consider configuring a separate LDAP server to run only the PC callable support. Because the server is not also running the backend controlling the data, fewer resources will be made non-swappable and the server will less likely need to be re-started. The disadvantage is that an extended operation request will require the LDAP server to communicate with another LDAP server for the data needed to satisfy the request, which can be slower than accessing that data on the same LDAP server. In general,

- If the data used in the extended operations is not on this system, then configure a separate LDAP server for the PC callable support.
- If the data is on this system, then try both configurations (PC callable support in a separate LDAP server and PC callable support in the same LDAP server as the data) to determine the impact on performance.

When using the PC callable support for RACF change logging, the LDAP server should also provide normal socket interfaces to allow usage of the change log entries.

At most, one LDAP server in a system can activate PC callable support. If an LDAP server tries to initialize PC callable support after another LDAP server has already tried (successfully or unsuccessfully) to initialize PC callable support, the initialization fails. The first LDAP server that tries to initialize the PC callable support locks the access to the PC callable support until that LDAP server has been shut down. If you are running LDAP in a sysplex, configure one LDAP server on each system in the sysplex to run the PC callable support. Each system should share the DB2 and RACF databases to ensure that they return the same results.

Setting up multiple LDAP servers

In order to set up two or more LDAP servers on the same system with different DB2-based backends, do the following:

- Follow the steps outlined in “Creating the DB2 database and table spaces for TDBM or GDBM” on page 42 and be sure to:
  1. Modify the SPUFI file that creates tables and indexes. Change `-UUUUUUUU-` (database owner) and `-DDDDDDDD-` (database name) to a different value and submit the SPUFI. A separate set of DB2 tables will be created.
  2. Update the configuration file **dbuserid** option with the value from step 1

Establishing the administrator DN and the replica server DN and passwords

There are several ways that the administrator DN and password or the replica server DN and password can be configured. One of these ways must be used, since an administrator DN and password are required for the LDAP server and some other LDAP programs to operate. The administrator DN must be present in the configuration file using the **adminDN** option (see page 56). The administrator DN password can optionally (this is not recommended) be placed in the configuration file using the **adminPW** option (see page 57) or can be held in the namespace managed by this instance of the LDAP server. If a replica is being established, the **masterServerDN** or **peerServerDN** option must be present in the configuration file. The **masterServerPW** or **peerServerPW** option can optionally be present. (This is also not recommended.) All of the options described below are applicable for **adminDN** and the first three options described below are applicable for **masterServerDN** and **peerServerDN**.

- Administrator DN and password in configuration file

  The simplest but least secure method is to select an administrator DN that is outside of the scope of suffixes managed by this server (see the **suffix** option on page 77). In other words, choose an administrator DN such that it does not fall within the portion or portions of the namespace managed by this server. Selection of this type of administrator DN requires that the password be placed in the configuration file using the **adminPW** option (see page 57).

  For example, you might choose a simple DN, such as `"cn=Admin"` for the administrator DN and a simple password such as `secret`. The configuration file options would then be established this way:

  ```
  adminDN "cn=Admin"
  adminPW secret
  ```

  Note: Do not use the example above without changing the password value, as well as the actual distinguished name.

  When a program or user binds using this administrator DN, the LDAP server verifies that the password supplied on the request matches the value provided in the configuration file for the **adminPW** option.

  Note: When first configuring a TDBM backend, it may be necessary to use this approach until the schema supporting the directory entries is loaded. Once the schema is loaded and the entry representing the administrator is added, the **adminDN** can be changed to the entry DN (see the

next list item regarding “Administrator DN and password as a TDBM entry”). The server must be restarted to pick up the new **adminDN**. Alternately, use the **ldif2tdbm** utility to load both the schema and **adminDN** entry.

- Administrator DN and password as a TDBM entry

  In this method, the administrator DN is established as an entry managed by the TDBM backend. The **userPassword** attribute is used to hold the password for the administrator DN in this case.

  For example, if the TDBM database is managing the portion of the namespace `"o=Your Company"`, one administrator DN that could be selected would be `"cn=LDAP Admin,o=Your Company"`.

  The configuration file would include the following options:

  ```
  adminDN "cn=LDAP Admin,o=Your Company"
  ...
  database tdbm GLDBTDBM
  ...
  suffix "o=Your Company"
  ```

  The LDIF-format entry to be added to the database through **ldapadd** or **ldif2tdbm** might be:

  ```
  dn: cn=LDAP Admin,o=Your Company
  objectclass: person
  cn: LDAP Admin
  description: Administrator DN for o=Your Company server
  sn: Administrator
  uid: admin
  userpassword: secret
  ```

  Note: Do not use the example above without changing the password value, as well as the actual distinguished name.

  If this entry is contained in an HFS file called `admin.ldif`, it can be loaded using **ldapadd**:

  ```
  ldapadd -h ldaphost -p ldapport -D binddn -w passwd -f admin.ldif
  ```

  Note that the LDIF-format entry can alternately be stored in a dataset and loaded through TSO using the sample CLIST for **ldapadd**.

  Note: The **ldapadd** example above assumes that the LDAP server is running and that the “suffix entry” (entry with the name `"o=Your Company"`) already exists. Furthermore, the *binddn* is assumed to exist and have sufficient authority to add the entry. When initially setting up the LDAP server, one way to satisfy the assumption is to first configure the LDAP server using the **adminDN** and **adminPW** configuration options. Then start the LDAP server, load the “suffix entry” and the “admin entry”, using the **adminDN** and **adminPW** configuration values for *binddn* and *passwd* respectively. After the add operations complete, stop the LDAP server, change the **adminDN** configuration option value to the name of the entry just added and **remove** the **adminPW** configuration option. Then restart the LDAP server.

  As an alternative to the steps listed in the previous paragraph, you can use the load utility (**ldif2tdbm** for TDBM) to load the admin DN entry.

  When a program or user binds using this administrator DN, the LDAP server verifies that the password supplied on the request matches the value of the **userPassword** attribute stored in the entry in DB2.

  CRAM-MD5 and DIGEST-MD5 authentication binds with the **adminDN** are supported as long as the entry exists in a TDBM backend. The **adminDN** entry must contain a **uid** attribute value that will be used as the user name by a client application when attempting a CRAM-MD5 or DIGEST-MD5 authentication bind. For more information on CRAM-MD5 and DIGEST-MD5 authentication, see Chapter 19, “CRAM-MD5 and DIGEST-MD5 Authentication,” on page 251.

- Administrator DN and password in RACF

  This method requires that the LDAP server be configured to use the RACF support provided in the SDBM backend. The administrator DN can be established as a RACF-style DN based upon a RACF user ID. (See “RACF-style distinguished names” on page 162 for more information.) In this case, the password for the administrator DN is the RACF user ID’s password, and is stored and verified by RACF.

For example, if you configure the LDAP server with RACF support where the portion of the namespace held by RACF is "sysplex=Sysplex1,o=Your Company", and the RACF user ID that is used for the administrator is g1admin, the configuration file would include these options:

```
adminDN "racfid=g1admin,profiletype=user,sysplex=Sysplex1,o=Your Company"
...
database sdbm GLDBSDBM
suffix "sysplex=Sysplex1,o=Your Company"
```

When a program or user binds using this administrator DN, the LDAP server makes a request to RACF to verify that the password supplied on the request matches the RACF password for RACF user ID g1admin.

Note that the user ID specified must have an OMVS segment defined and an OMVS UID present.

- **krbLDAPAdmin** in the configuration file

  You may wish to configure the administrator to be able to bind to the server through Kerberos. In this case you need to create a Kerberos identity for the user and also add this value to the configuration file. You cannot use the **krbLDAPAdmin** option to provide a Kerberos identity for the **masterServerDN** or **peerServerDN** option.

  For example, if the user ID for the LDAP administrator is **LDAPADM** and this Kerberos identity was configured to be ldapadm@realm1.com then your configuration file will have the following:

  ```
  krbLDAPAdmin ibm-kn=ldapadm@realm1.com
  ```

  This allows the administrator to bind to the server through Kerberos Authentication rather than by performing a simple bind.

Example configuration scenarios

This section shows scenarios of LDAP server configurations.

Configuring a TDBM backend with SSL/TLS only and password encryption

The configuration example in this section uses the TDBM backend and shows the configuration file checklist next to the corresponding sample configuration file.

Table 10 on page 54 shows all the options that are available for each section.

Table 13. Sample checklist and slapd.conf (using TDBM, SSL/TLS, and password encryption)

Section	Check	Sample slapd.conf

Global	√	`# Filename slapd.conf`
		
		`# Global section`
		`sizelimit 500`
		`timelimit 3600`
		`adminDn "cn=LDAP Administrator,o=Your Company"`
		
		`listen ldaps://:636`
		`sslAuth serverClientAuth`
		`sslCertificate none`
		`sslCipherSpecs 15104`
		`sslKeyRingFile /u01/ldapsrv/ldapsrv.kdb`
		`sslKeyRingPWStashFile /u01/ldapsrv/ldapsrv.sth`
		
		`# TDBM backend section`
		`database tdbm GLDBTDBM LocalDirectory`
		`suffix "o=Your Company"`
		`servername LOC1`
		`dbuserid LDAPSRV`
		`databasename LDAPDB`
		`attrOverflowSize 500`
		`pwEncryption MD5`

SSL/TLS	√	
Sysplex		
Kerberos		
SDBM backend		
Kerberos		
GDBM backend		
Multiserver		
TDBM backend	√	
Password encryption	√	
Replication		
Multi-server		
Kerberos		
Native authentication		
EXOP backend		

Configuring SDBM and GDBM backends

The configuration example in this section uses SDBM and GDBM backends and shows the configuration file checklist next to the corresponding sample configuration file.

Table 14. Sample checklist and slapd.conf (using SDBM and GDBM)

Section	Check	Sample slapd.conf

Global	√	`# Filename slapd.conf`
		
		`# Global section`
		`sizelimit 500`
		`timelimit 3600`
		`adminDn "racfid=ldadmin,profiletype=user,cn=myRACF"`
		`listen ldap://:pc`
		`listen ldap://:389`
		
		`# SDBM backend section`
		`database sdbm GLDBSDBM`
		`suffix "cn=myRACF"`
		
		`# GDBM backend section`
		`database gdbm GLDBGDBM`
		`servername LOC1`
		`dbuserid LDAPSRV`
		`attrOverflowSize 500`

SSL/TLS		
Sysplex		
Kerberos		
SDBM backend	√	
Kerberos		
GDBM backend	√	
Multi-server		
TDBM backend		
Password encryption		
Replication		
Multi-server		
Kerberos		
Native authentication		
EXOP backend		

Configuring SDBM and TDBM backends

The configuration example in this section uses both SDBM and TDBM backends and shows the configuration file checklist next to the corresponding sample configuration file.

Table 10 on page 54 shows all of the options that are available for each section.

Table 15. Sample checklist and slapd.conf (using SDBM and TDBM)

Section	Check	Sample slapd.conf

Global	√	`# Filename slapd.conf`
		
		`# Global section`
		`sizelimit 500`
		`timelimit 3600`
		`adminDn "racfid=ldadmin,profiletype=user,cn=myRACF"`
		
		`# SDBM backend section`
		`database sdbm GLDBSDBM`
		`suffix "cn=myRACF"`
		
		`# TDBM backend section`
		`database tdbm GLDBTDBM`
		`suffix "o=Your Company"`
		`servername LOC1`
		`dbuserid LDAPSRV`
		`databasename LDAPDB`
		`attrOverflowSize 500`

SSL/TLS		
Sysplex		
Kerberos		
SDBM backend	√	
Kerberos		
GDBM backend		
Multi-server		
TDBM backend	√	
Password encryption		
Replication		
Multi-server		
Kerberos		
Native authentication		
EXOP backend		

Configuring an EXOP backend

The configuration example in this section uses an EXOP backend and shows the configuration file checklist next to the corresponding sample configuration file.

Table 10 on page 54 shows all of the options that are available for each section.

Table 16. Sample checklist and slapd.conf (using EXOP)

Section	Check	Sample slapd.conf

Global	√	`# Filename slapd.conf`
		
		`# Global section`
		`listen ldap://:pc`
		
		`# EXOP backend section`
		`database exop GLDXPDIR`

SSL/TLS		
Sysplex		
Kerberos		
SDBM backend		
Kerberos		
GDBM backend		
Multi-server		
TDBM backend		
Password encryption		
Replication		
Multi-server		
Kerberos		
Native authentication		
EXOP backend	√	

Configuring and using multiple concurrent servers in a sysplex

Following is a simplified example scenario to demonstrate the configuration and usage of the multi-server operation mode with dynamic workload balancing enabled. While the user's operational environment may be more complex than this example, the lessons demonstrated apply in a similar fashion.

See "Determining operational mode" on page 82 for more information about operating in a sysplex.

Scenario

ABC Company is running a 3-way Parallel Sysplex with DB2 data sharing available on each host in the sysplex. Configure an instance of the LDAP server on each of the z/OS systems in the sysplex, serving the same LDAP directory TDBM database, such that the three instances are functional equivalents of each other, permitting users to exploit dynamic workload balancing across these LDAP servers in the sysplex.

Assume that at this point, the system administrator has installed and configured a DB2 subsystem on each host in the sysplex to be part of the same data sharing group, and that the system administrator has configured at least one Domain Name Service (DNS) server for the sysplex. In addition, the TCP/IP stacks which serve each host are registered with Workload Manager (WLM). See "Dependencies" on page 85 for more information.

The operational environment in which the LDAP servers will run is configured as indicated in the following diagram:

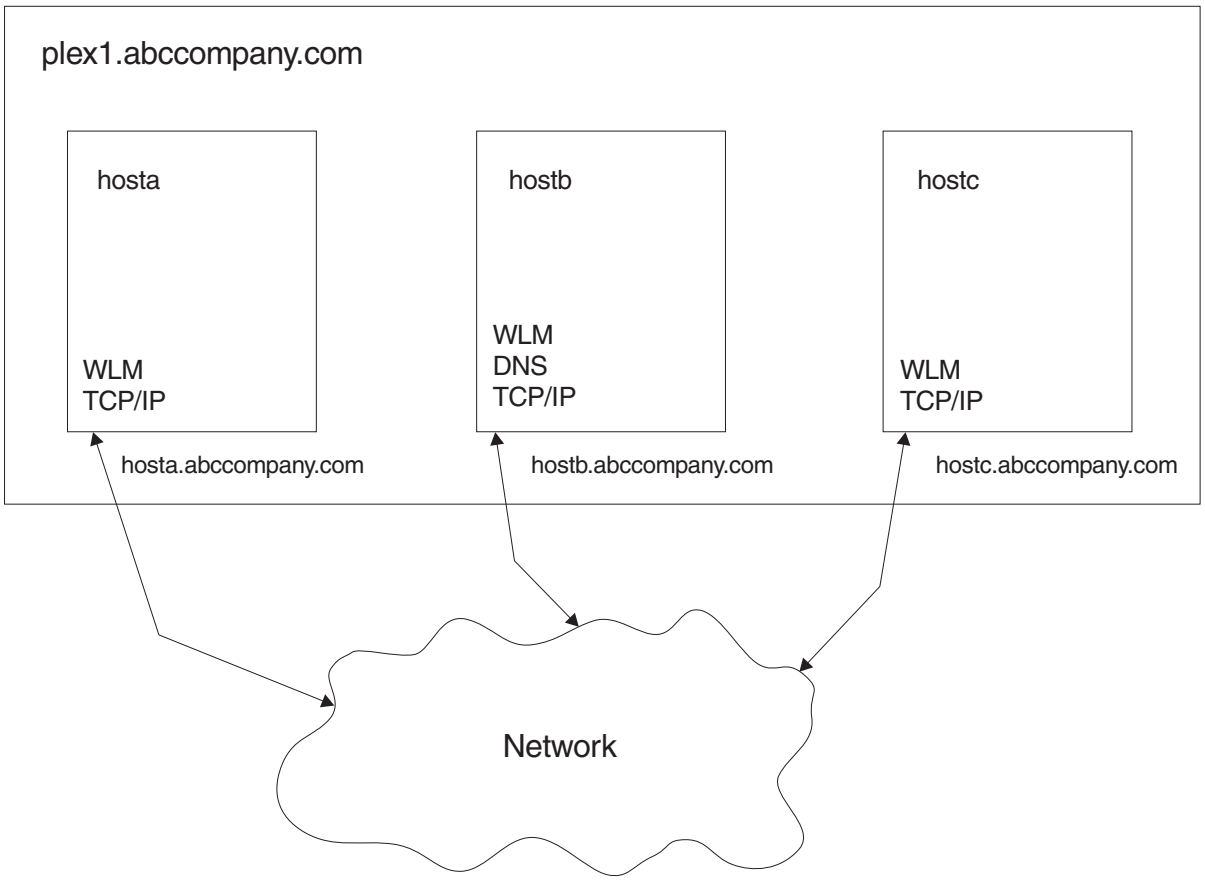


Figure 7. Multi-server sample configuration (phase 1)

The three host systems are named `hosta`, `hostb`, and `hostc`, and are coupled in sysplex `plex1` in Internet sub-domain `abccompany.com`. Each of the host systems is configured with a single network adapter, and these systems are accessible with these names:

```
hosta.abccompany.com
hostb.abccompany.com
hostc.abccompany.com
```

The sysplex domain name for this sysplex is `plex1.abccompany.com`. The primary DNS server for the sysplex domain runs on `hostb`.

Three server instances will be started, one on each host in the sysplex. A checklist and the server configuration file for each of the three servers follows and differences among the files are highlighted.

Table 17. Sample checklist (using TDBM with multi-server and dynamic workload management)

Section	Check
Global	√
SSL/TLS	
Sysplex	√
Kerberos	
SDBM backend	
Kerberos	
GDBM backend	
Multi-server	
TDBM backend	√
Password encryption	
Replication	
Multi-server	√
Kerberos	
Native authentication	

```
# * Filename slapd.conf

listen ldap://:389
listen ldaps://:636
commThreads        60
maxconnections     40000
timelimit          3600
sizelimit          500
sysplexGroupName  ldapgrp1
sysplexServerName servera
#######################################################################
# tdbm database definitions
#######################################################################

database      tdbm GLDBTDBM
suffix        "o=Your Company"

multiserver   y
readOnly      off

# The following options must be filled in with appropriate values
# for your DB2 setup, prior to attempting to run with the DB2 backend.

servername    loc1
databasename  abcdb1
dbuserid      dbu01
dsnaoini ABCCO.DB2CLI.CLIINIA
```

Figure 8. Configuration file for Server A on hosta

Figure 9 shows the contents of ABCCO.DB2CLI.CLIINIA:

```
;This is a comment line...
; Example COMMON stanza
[COMMON]
MVSDEFAULTSSID=DB1G

; Example SUBSYSTEM stanza for your DB2 subsystem name
[DB1G]
MVSATTACHTYPE=RRSAF
PLANNAME=DSNACLI

; Example DATA SOURCE stanza for your data source
[LOC1]
AUTOCOMMIT=0
CURSORHOLD=0
CONNECTTYPE=1
```

Figure 9. Contents of ABCCO.DB2CLI.CLIINIA

```
# * Filename slapd.conf

listen ldap://:389
listen ldaps://:636
commThreads        60
maxconnections     40000
timelimit         3600
sizelimit         500
sysplexGroupName  ldapgrp1
sysplexServerName serverb

######################################################################
# tdbm database definitions
######################################################################

database     tdbm GLDBTDBM
suffix       "o=Your Company"

multiserver        y
readOnly           off

# The following options must be filled in with appropriate values
# for your DB2 setup, prior to attempting to run with the DB2 backend.

servername         loc1
databasename       abcdb1
dbuserid           dbu01
dsnaoini ABCCO.DB2CLI.CLIINIB
```

Figure 10. Configuration file for Server B on hostb

Figure 11 shows the contents of ABCCO.DB2CLI.CLIINIB:

```
;This is a comment line...
; Example COMMON stanza
[COMMON]
MVSDEFAULTSSID=DB2G

; Example SUBSYSTEM stanza for your DB2 subsystem name
[DB2G]
MVSATTACHTYPE=RRSAF
PLANNAME=DSNACLI

; Example DATA SOURCE stanza for your data source
[LOC1]
AUTOCOMMIT=0
CURSORHOLD=0
CONNECTTYPE=1
```

Figure 11. Contents of ABCCO.DB2CLI.CLIINIB

```
# * Filename slapd.conf

listen ldap://:389
listen ldaps://:636
commThreads        60
maxconnections     40000
timelimit         3600
sizelimit         500
sysplexGroupName  ldapgrp1
sysplexServerName serverc

#####################################################################
# tdbm database definitions
#####################################################################

database        tdbm GLDBTDBM
suffix          "o=Your Company"

multiserver  y
readOnly     off

# The following options must be filled in with appropriate values
# for your DB2 setup, prior to attempting to run with the DB2 backend.

servername   loc1
databasename abcdb1
dbuserid     dbu01
dsnaoini ABCCO.DB2CLI.CLIINIC
```

Figure 12. Configuration file for Server C on hostc

Figure 13 shows the contents of ABCCO.DB2CLI.CLIINIC:

```
;This is a comment line...
; Example COMMON stanza
[COMMON]
MVSDEFAULTSSID=DB3G

; Example SUBSYSTEM stanza for your DB2 subsystem name
[DB3G]
MVSATTACHTYPE=RRSAF
PLANNAME=DSNACLI

; Example DATA SOURCE stanza for your data source
[LOC1]
AUTOCOMMIT=0
CURSORHOLD=0
CONNECTTYPE=1
```

Figure 13. Contents of ABCCO.DB2CLI.CLIINIC

The DB2 subsystem IDs must be different on each system and those subsystem IDs must all be defined into the same DB2 data sharing group.

Several things should be noted in the server configuration files above:

- All three configuration files use the same TDBM database (which is accessible through a DB2 data sharing group which contains the subject database) and database resources, as indicated by identical values for the parameters **servername**, **databasename**, and **dbuserid**. Each server configuration file points to the DB2 Call Level Interface (CLI) initialization file for that respective server. The MVSDEFAULTSSID defined in the CLI initialization file for each server must be the DB2 subsystem name or the DB2 group attachment name on the host on which that server instance will be started which is a member of the DB2 data sharing group which all three systems in the sysplex share, and which contains the TDBM database of interest.
- All three configuration files use the same name for their **sysplexGroupName** option; this is required to ensure all three servers are recognized as functional equivalents of each other for purposes of dynamic workload management. In addition, the **sysplexServerName** for each of the three must be unique within this **sysplexGroupName** in this sysplex.
- The ports (both secure and nonsecure) on the **listen** parameters in the configuration file must be the same for all servers in the same **sysplexGroupName** for the dynamic workload management to be effective.

With the additional configuration information just outlined, we can extend the diagram in Figure 7 on page 93 to look like this:

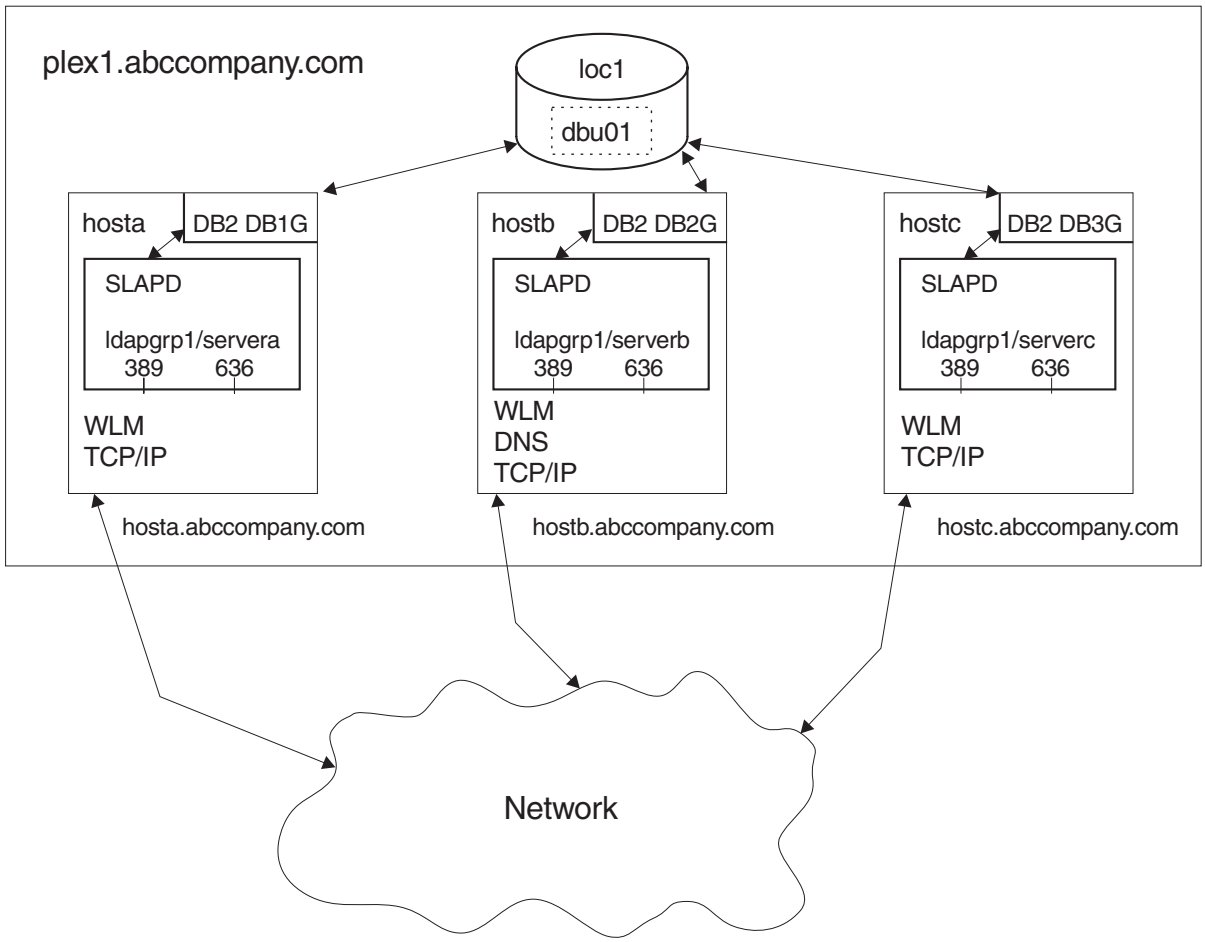


Figure 14. Multi-server sample configuration (phase 2)

When the LDAP servers are started, the following messages will be printed to STDERR:

```
GLD0115I Workload Manager enablement initialization successful for
 group=ldapgrp1, server=servera on host HOSTA.
```

```
GLD0115I Workload Manager enablement initialization successful for
 group=ldapgrp1, server=serverb on host HOSTB.

GLD0115I Workload Manager enablement initialization successful for
 group=ldapgrp1, server=serverc on host HOSTC.
```

At this point, a client can send a search request to any LDAP server instance which will provide service for the TDBM database of interest by using the sysplex group name in the DNS host name option as in:

```
ldapsearch -h ldapgrp1.plex1.abccompany.com -p 389 -D "cn=admin" -w secret
 -b "o=Your Company" objectclass=person cn
```

With the host specified as `ldapgrp1.plex1.abccompany.com`, the request will be routed to the server instance in **sysplexGroupName** `ldapgrp1` in sysplex `plex1.abccompany.com` which has the most available resources with which to perform the work (see Figure 14 on page 98). Although requests could be directed at a specific server instance (that is, specifying `serverc.ldapgrp1.plex1.abccompany.com` or `hostc.abccompany.com` for the DNS host name), doing so defeats the use of the dynamic workload management features by bypassing the TCP/IP connection optimization.

See *z/OS Integrated Security Services LDAP Client Programming* for more information about **ldapsearch**.

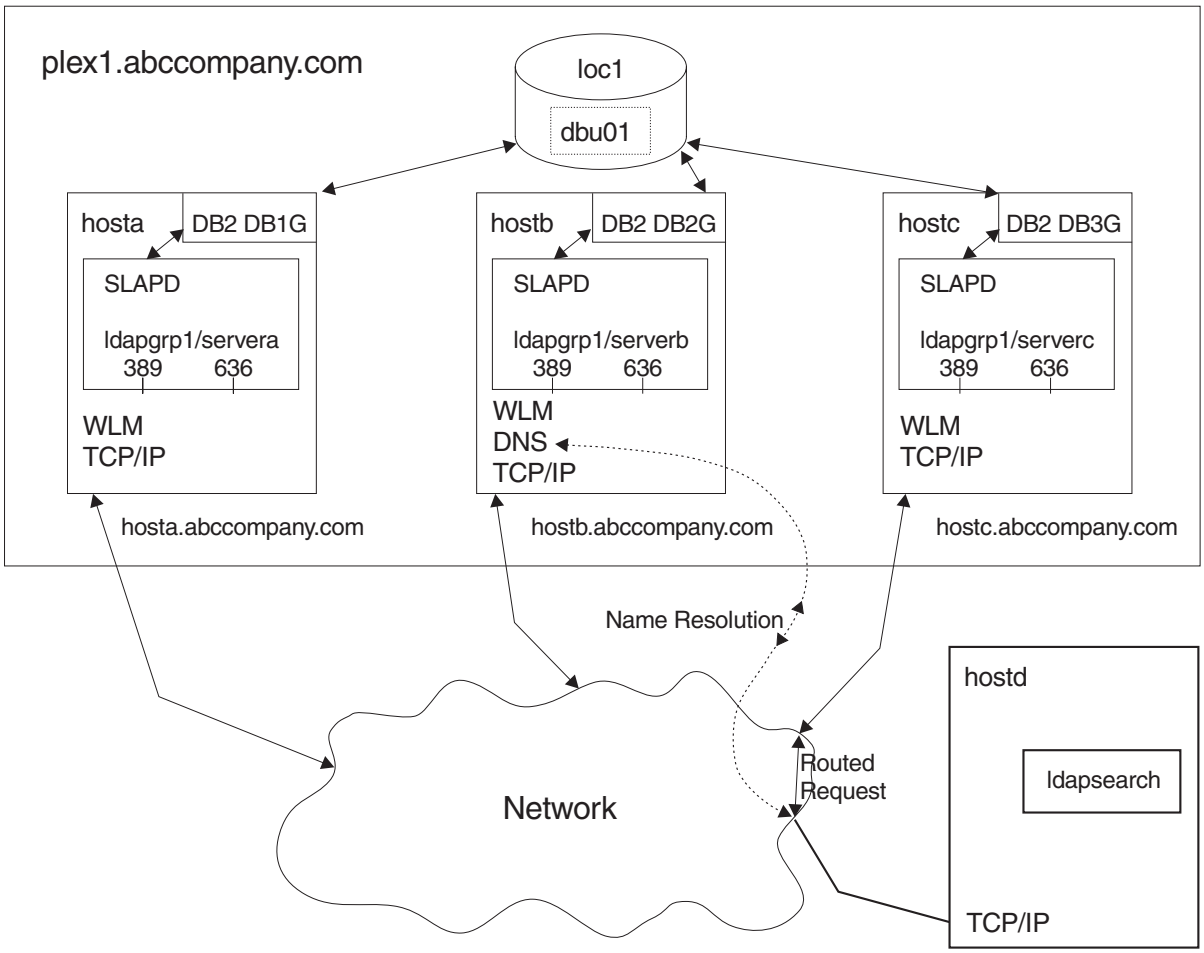


Figure 15. Multi-server sample configuration (phase 3)

It should be noted that for proper workload management using TCP/IP connection optimization, DNS queries must be answered by the name server within the sysplex. For this reason, name servers that are located outside the sysplex cannot be configured as primary or secondary servers for the sysplex domain. Thus, in this example, `hostd` must be configured to resolve names through the sysplex name server.

Chapter 9. Running the LDAP server

This chapter describes what is necessary to get the LDAP server running.

Setting up the PDS for the LDAP server DLLs

The LDAP server searches for and loads a number of dynamic load libraries (DLLs) during its startup processing. All DLLs for the LDAP server are shipped in PDS format only. In order for these DLLs to be located by the LDAP server at runtime, the PDS which contains these DLLs (SYS1.SIEALNKE) must either be in the LINKLIST (the default installation), referenced in a **STEPLIB DD** card, if the LDAP server is started from JCL, or listed in the **STEPLIB** environment variable, if the LDAP server is started from the z/OS UNIX System Services command prompt. Any of these methods can be used, and the choice of the best method is dependent on the way you will most often be running the LDAP server. If you put SYS1.SIEALNKE in LINKLIST, **STEPLIB** is not necessary.

The LDAP server also depends on the **SCEERUN** dataset. Add **SCEERUN** to your LINKLIST or, if that is not possible, add it to **STEPLIB**. See *z/OS Program Directory* for more information.

Setting up and running the LDAP server as a started task

To run the LDAP server as a started task, you must define the started task for the LDAP server and then you can run the LDAP server using JCL.

Defining the started task for the LDAP server

After you create the LDAPSRV user ID (described in “Requirements for a user ID that runs the LDAP server” on page 37), you must define the LDAPSRV started task. The examples and the sample startup procedure use the name LDAPSRV for this task, but you can use any name for it.

To define the started task for the user ID you just created, you can use the following RACF commands.

```
RDEFINE STARTED LDAPSRV.** STDATA(USER(LDAPSRV))
SETROPTS RACLIST(STARTED) REFRESH
```

Running the LDAP server using the sample JCL

The JCL needed to run the LDAP server as a started task is provided with the product as a procedure. This JCL can be found in *GLDHLQ*.SGLDSAMP on the system where the LDAP server is installed. If you have a ServerPac installation, *GLDHLQ* will be **GLD**. This JCL procedure can be started in the System Display and Search Facility (SDSF) or from the operator’s console, once the sample JCL has been placed into the installation-specific library for procedures. This JCL must be tailored before it can be run.

To start the LDAP server in SDSF, enter:

```
/s ldapsrv
```

To start the LDAP server from the operator’s console, enter:

```
s ldapsrv
```

The LDAP server has the following optional command-line parameters. One or more of these may be specified when starting the LDAP server.

-f *pathname*
: Name of configuration file to be read. Default is **/etc/ldap/slapd.conf**.

-l *ldap_URL*
: Host name or IP address and port number on which the LDAP server should bind and listen for incoming requests. Refer to the **listen** parameter on page 65 for information on the *ldap_URL*

parameter. The **-l** parameter can be specified multiple times to add additional *ldap_URL* values. The values specified using the **-l** command-line parameter override the values specified for the **listen** option in the configuration file.

It is advisable to reserve the port number or numbers chosen here in your TCP/IP profile data set. Also, be aware that port numbers below 1024 may require additional specification. See *z/OS Communications Server: IP Configuration Guide*.

-m Start the server in maintenance mode rather than normal mode. Maintenance mode restricts the directory updates processed by the server. See “Maintenance mode” on page 296 for more information.

-p *port*

Note: The port option is deprecated when the **listen** option is specified. See page 65 for information on the **listen** option.

Port number where the LDAP server will listen for nonsecure communications. If you specify this parameter, it overrides the value that may be in the **slapd.conf** configuration file if there are not any **listen** parameters specified in the configuration file or on the command line.

-s *secureport*

Note: The port option is deprecated when the **listen** option is specified. See page 65 for information on the **listen** option.

Port number where the LDAP server will listen for secure communications. If you specify this parameter, it overrides the value that may be in the **slapd.conf** configuration file if there are not any **listen** parameters specified in the configuration file or on the command line.

-d *debug_level*

The debug level is a mask that may be specified as follows:

- A decimal value (for example, 32)
- A hexadecimal value (for example, x20 or X20)
- A keyword (for example, FILTER)
- A construct of those values using plus and minus signs to indicate inclusion or exclusion of a value. For example:
  - ’32768+8’ is the same as specifying ’32776’, or ’x8000+x8’, or ’ERROR+CONNS’
  - ’2146959359’ is the same as specifying ’ANY-STRBUF’
  - By beginning the debug level with a minus sign, you can deactivate debug collection for the various types. ″-CONNS″ modifies an existing debug level by deactivating connection traces.
  - By beginning the debug level with a plus sign, you can activate debug collection for the various types. ″+CONNS″ modifies an existing debug level by activating connection traces.

Table 18 lists the debug levels and the related decimal, hexadecimal, and keyword values.

Table 18. Debug levels

Keyword	Decimal	Hexadecimal	Description
OFF	0	0x00000000	No debugging
TRACe	1	0x00000001	Entry and exit from routines
PACKets	2	0x00000002	Packet activity
ARGS	4	0x00000004	Data arguments from requests
CONNs	8	0x00000008	Connection activity
BER	16	0x00000010	Encoding and decoding of data, including ASCII and EBCDIC translations, if applicable

Table 18. Debug levels (continued)

Keyword	Decimal	Hexadecimal	Description
FILTer	32	0x00000020	Search filters
MESSage	64	0x00000040	Messaging subsystem activities and events
ACL	128	0x00000080	Access Control List activities
STATs	256	0x00000100	Operational statistics
THREad	512	0x00000200	Threading activities
REPLication	1024	0x00000400	Replication activities
PARSe	2048	0x00000800	Parsing activities
PERFormance	4096	0x00001000	Relational backend performance statistics
Reserved	8192	0x00002000	Reserved
REFErral	16384	0x00004000	Referral activities
ERROr	32768	0x00008000	Error conditions
SYSPlex	65536	0x00010000	Sysplex/WLM activities
MULTIServer	131072	0x00020000	Multi-server activities
LDAPBE	262144	0x00040000	Connection between a frontend and a backend
STRBuf	524288	0x00080000	UTF-8 support activities
TDBM	1048576	0x00100000	Backend activities (TDBM)
SCHEma	2097152	0x00200000	Schema support activities (TDBM)
BECApabilities	4194304	0x00400000	Backend capabilities
CACHe	8388608	0x00800000	Cache activities
ANY	2147483647	0x7FFFFFFF	All levels of debug
ALL	2147483647	0x7FFFFFFF	All levels of debug
Note: The minimum abbreviation for each keyword is shown in uppercase letters.			

The debug level for the server can be set at a number of different times.

- The initial debug level is OFF.
- Prior to starting the server, the **LDAP_DEBUG** environment variable may be set. The server uses this value first. For example,

  ```
  export LDAP_DEBUG='ERROR+TRACE'
  ```

- When starting the server, the **-d** parameter may be specified. The debug level specified on this parameter either replaces, adds to or deletes from the preceding debug level. For example,
  - `s ldapsrv,parms='-d ERROR'`

    replaces the current debug level that is either off or has been set by the **LDAP_DEBUG** environment variable with the new debug level of only ERROR.
  - `s ldapsrv,parms='-d +ERROR'`

    adds the ERROR debug level to the current debug level that is either off or has been set by the **LDAP_DEBUG** environment variable.
  - `s ldapsrv,parms='-d -ERROR'`

    removes the ERROR debug level from the current debug level that is either off or has been set by the **LDAP_DEBUG** environment variable.
- It is possible to change the debug level while the server is running whether it was started as a started task or from the shell. See “Dynamic debugging” on page 106 for more information.

All informational and error messages are printed to the started task output for the LDAP server.

When the LDAP server has been started and is ready, the message

```
GLD0122I SLAPD is ready for requests.
```

is displayed.

"Running the LDAP server using data sets" discusses using a data set for the configuration file. In order to specify the configuration file as either a data set name or a DD name in SDSF, some special syntax is necessary.

In order to specify a full data set name, it may be necessary to be in the expanded input screen for SDSF. This is accomplished by entering a slash (/) in SDSF. On the expanded screen, it is then possible to specify a data set name for the configuration file. Assuming that the configuration file has been established in data set `MYUSERID.SLAPD.CONF`, the start command for the LDAP server in expanded input SDSF or the console would be:

```
s ldapsrv,parms='-f //''''MYUSERID.SLAPD.CONF'''''
```

or, if additional parameters are desired:

```
s ldapsrv,parms='-f //''''MYUSERID.SLAPD.CONF'''' -l ldap://:999'
```

If a **DD** name, **SLAPCONF**, was established in the **LDAPSRV PROC**, as follows:

```
SLAPCONF DD DSN=MYUSERID.SLAPD.CONF,DISP=SHR
```

the LDAP server could be started from expanded input SDSF or the console by entering:

```
s ldapsrv,parms='-f //DD:SLAPCONF'
```

To stop the LDAP server in SDSF, enter:

```
/p ldapsrv
```

To stop the LDAP server from the operator's console, enter:

```
p ldapsrv
```

This command causes the LDAP server to shut down.

Started task messages

The LDAP server writes messages to **stdout** and **stderr**. Messages sent to **stdout** and **stderr** appear in **DD:SLAPDOUT** in the provided JCL when running the LDAP server as a started task. **SLAPDOUT** appears in the started task log for the LDAP server and can be viewed through SDSF. See *z/OS SDSF Operation and Customization* for information on how to use SDSF.

Running the LDAP server using data sets

Note: Using the LDAP configuration utility (**ldapcnf**) to configure your server creates all the necessary files in a partitioned data set.

The LDAP server, when run as a started task, accepts several of its files as data sets. Data set versions of the configuration files and **envvars** file are not shipped with the LDAP server, but can be created using the **OGET** command to copy the HFS versions of the files into data sets. (See *z/OS DCE Command Reference* for information on the use of the **OGET** command.)

The default data set characteristics for record format and record length (V 255) which **OGET** will use when creating a new data set are not acceptable for JCL when submitting for batch processing. In order to avoid this, allocate the MYUSER.DSNTIJCL sequential data set to be fixed block 80 prior to performing the **OGET** operation.

A data set version of the DSNAOINI file needed for the TDBM backend can be created by copying and editing the default file provided by DB2. See step 4 on page 14. The DSNAOINI file can be specified either in the configuration file or in a **DSNAOINI DD** statement, or a DSNAOINI environment variable can be used. The **DD** statement takes precedence.

Note: Be sure that use of sequence numbers is turned off when editing this dataset.

Once the data set versions of these files are available, they can be specified in the LDAPSRV procedure. The configuration file can be specified using the **CONFIG DD** statement, the **envvars** file can be specified using the **ENVVAR DD** statement, and the **DSNAOINI** file can be specified using the **DSNAOINI DD** statement.

Setting up and running the LDAP server in the z/OS shell

A sample shell script is shipped for running the LDAP server in the shell. The sample shell script is located in **/usr/lpp/ldap/sbin/slapd**. The shell script needs to be modified to fit your environment. The **PATH** variable should be set in the shell script. Ensure that **/usr/sbin** is added to the **PATH** environment variable. Also, set **STEPLIB** to SYS1.SIEALNKE if the PDS has not been placed in LINKLIST. If your LDAP server is configured to encrypt passwords, the **LIBPATH** variable may need to be set. See "Installing OCSF and ICSF for password encryption" on page 16 for more information. The LDAP server writes messages and debugs to **stdout** and **stderr**. It is recommended that **stdout** and **stderr** be redirected and saved to capture any messages. Then, start the LDAP server by issuing:

```
slapd >/tmp/slapd.log 2>&1 &
```

The same parameters described in "Setting up and running the LDAP server as a started task" on page 101 can be provided to the LDAP server when starting it from the z/OS shell.

It is also possible to define a separate user ID that will be used to run the LDAP server. See "Requirements for a user ID that runs the LDAP server" on page 37 for instructions on defining a separate user ID to run the LDAP server.

When started, the LDAP server will read an environment variable file. The default file is **/etc/ldap/slapd.envvars**. This default can be changed by setting the environment variable **LDAP_SLAPD_ENVVARS_FILE** to the full path name of the desired environment variable file.

To stop the LDAP server in the z/OS shell, it is necessary to know its process ID. On any user ID that has a UID of 0, enter:

```
ps -ef | grep slapd
```

This will provide the process ID for the LDAP server. Next, enter:

```
kill -15 process-ID
```

where *process-ID* is the process ID of the LDAP server from the **ps -ef** command. This command will cause the LDAP server to shut down.

If multiple LDAP servers are running on the system, be sure to pick the correct server's process ID before entering the **kill** command.

Verifying the LDAP server

The following examples show how you can verify your LDAP server using the **ldapsearch** tool. Note that you can use any LDAP client to do this.

- **Verifying TDBM**

  In the command below, substitute the suffix value from your configuration file for the **-b** parameter. The command can be run multiple times to verify each suffix defined in the configuration file.

  ```
  ldapsearch -h 127.0.0.1 -s base -b "o=Your Company" "objectclass=*"
  ```

  Note: The LDAP search will return the message ″No such object″ if the suffix entries have not been loaded into the directory.

- **Verifying SDBM**

  For SDBM, you must bind with a valid RACF-style DN to perform the search. Substitute a RACF ID of your choice in the `racfid` portion of the DN on the **-D** and the **-b** parameters below. Also, substitute your SDBM suffix in the DN on the **-D** and **-b** parameters. The RACF password for the user ID used in the **-D** parameter must be specified in the **-w** parameter.

  ```
  ldapsearch -h 127.0.0.1 -D racfid=IBMUSER,profiletype=user,cn=myRacf
    -w password_for_IBMUSER -b racfid=IBMUSER,profiletype=user,cn=myRacf "objectclass=*"
  ```

- **Verifying GDBM**

  For GDBM, you must bind with the LDAP administrator DN or another DN authorized to search the change log.

  ```
  ldapsearch -h 127.0.0.1 -D bindDn -w bindPw -s base -b cn=changelog "objectclass=*"
  ```

The previous **ldapsearch** examples assume a default port of 389. If your port is not 389, use the **-p** parameter to specify the correct port.

Be sure to substitute the correct TCP/IP host name or TCP/IP address for the `127.0.0.1` after the **-h** parameter. The **-b** parameter specifies the starting point for the search. The use of the quotation marks around the filter prevents the asterisk (*) from being interpreted by the shell.

Note that this can be done from TSO as well, substituting **LDAPSRCH** for **ldapsearch**.

See *z/OS Integrated Security Services LDAP Client Programming* for more information about **ldapsearch**.

DB2 and TCP/IP termination

The LDAP server uses DB2 to store the directory information for the TDBM and GDBM backends. When one of these backends is configured, the LDAP server monitors DB2 and detects when it shuts down. Based on the **db2terminate** configuration option, the LDAP server can then either shut itself down, or continue running without access to the TDBM and GDBM backends and reconnect to DB2 when it becomes available. See page 60 for the description of the **db2terminate** configuration option.

The LDAP server uses TCP/IP for its client communications, except for applications using the LDAP Program Call support. The LDAP server also monitors TCP/IP and detects when it shuts down. In this case, the LDAP server shuts itself down. There is no configuration option to change this behavior. You should restart TCP/IP and then LDAP.

Dynamic debugging

When the LDAP server is running as a started task or from the z/OS shell, it is possible to dynamically turn the debugging facility on and off. You can also replace the current debug levels, add to the current debug levels, or remove from the current debug levels. The following command can be sent to the LDAP server from the SDSF or the operator's console. Note that if the command is entered from SDSF, it must be preceded by a slash (/). In the command:

```
f ldapsrv,appl=debug=debug_level
```

the *debug_level* is a mask that specifies the desired debug level. (See Table 18 on page 102 for more information.) To send the same command to the LDAP server in the z/OS shell, it is necessary to know the job name assigned to the process by performing

```
d a,l
```

from SDSF or the operator's console and determining the name, which includes the user ID under which the LDAP server is running and a suffix. Once this name is found, use it to replace ldapsrv in the command above. See Table 18 on page 102 for an explanation of the debug level values.

Debug information will be added to the output associated with the LDAP server.

To turn the debug tracing off, enter the same command providing the value zero (0) for *debug_level*.

Gathering trace records into the in-storage trace table

In addition to debugging, the LDAP server also keeps an internal trace table. A set of trace records is stored automatically (by default) in the in-storage trace table. Other traces, in addition to the default trace records, are controlled by a debug file. Use the **_GLD_DEBUGFILE_NAME** environment variable to specify the name of the debug file. The debug file contains entries which indicate what optional traces to include in the trace table. The size of the in-storage trace table can be set using the **_GLD_TRACE_TABLE_SIZE** environment variable. The default size is 512K. You can specify the value in K, M, or bytes (for example, `1024K`, `1M`, `1048577`). Note that once the trace table is full, the records begin to wrap (that is, the old records are overwritten by the new records).

You have the following options when gathering trace records for the LDAP server:
- Determining which trace records go into the in-storage trace table
- Printing the trace table
- Resetting the trace table

The command:

```
f ldapsrv,appl=debug
```

reads the debug file for the internal trace and resets all trace settings based on what is in the debug file. (Note that if the command is entered from SDSF, it must be preceded by a slash (/).) To send the same command to the LDAP server in the z/OS shell, it is necessary to know the job name assigned to the process. Do this by performing

```
d a,l
```

from SDSF or the operator's console and determining the name, which includes the user ID under which the LDAP server is running and a suffix. Once this name is found, use it to replace ldapsrv in the command above.

Customizing the trace table

You can customize the trace entries produced in the in-storage trace table using the **gldddebug** statements in the debug file. The **gldddebug** statement has the following format:

gldddebug=(c=*type***,p=***prt***,l=***tLvl***)***cmt*

where:

type Is the trace type. Choose one of the following for *type*:
- **STORage**
- **LOCK**
- **ASYNCio**

- **LDAPDebug**
- **ALL**

The uppercase shows the minimum abbreviation that can be used.

prt Is the output direction. Choose one of the following for *prt*:
- **m** - same place as in-storage trace
- **y** - print to stderr
- **n** - nothing in output

tLvl Is the trace level from 0-127. Each trace is identified by a type and level. The higher the number, the more traces that can be selected. For example, if you specify 120, 0-120 are selected.

cmt Is a comment in the form:

```
/* your comment */
```

Following is an example of using **glddebug**:

```
glddebug=(c=stor,p=m,l=127)
```

This statement would result in sending the output of all storage traces to the in-storage trace table.

Printing the trace table

Use the **_GLD_TRACEFILE_NAME** environment variable to specify the name of the file to which the trace table is printed. The file name defaults to **/etc/ldap/gldtrace.output**. If the trace file will not open, the trace file goes to stderr. Also, if there is an abend, the trace table is printed automatically.

To print the trace table, enter the following modify command:

```
f ldapsrv,appl=trace,print
```

The messages below are displayed on an error case and printed to the in-memory trace table which will be printed to the file specified on a server abend or when the user prints the table.

The following is an example of in-memory output:

```
do_search bind=dn entry=dn filter=* scope=0 rc=1
do_search msg=error msg related to the problem here like debug error msg
```

Where scope is

```
baseObject = 0, singleLevel = 1, wholeSubtree = 2.
```

```
do_modify bind=dn entry=dn op=0 rc=1
do_modify msg=error msg related to the problem here like debug error msg
```

Where op is

```
add = 0, replace = 1, delete = 2.
```

```
do_modrdn bind=dn entry=dn newrdn=dn newSuperior=dn rc=1
do_modrdn msg=error msg related to the problem here like debug error msg

do_add bind=dn entry=dn rc=1
do_add msg=error msg related to the problem here like debug error msg
```

The remaining operations will have the same information as the do_add operation above.

Resetting the trace table

To reset the trace table (clear the trace table of its contents), enter the following modify command:

```
f ldapsrv,appl=trace,reset
```

Interaction between debug and in-storage trace

The debugging facility and the in-storage trace table interact as follows:

- The command

  ```
  f ldapsrv,appl=debug=debug_level
  ```

  controls debug tracing. The command

  ```
  f ldapsrv,appl=debug
  ```

  (without the "=*debug_level*"), reads the debug file to start the internal trace.
- A *type* setting of **LDAPDebug** on the **glddebug** statement directs debugs to the internal trace table.

Capturing performance information

The LDAP server provides a **query** command that is used to capture various performance information. The output of the **query** command goes to the system log and the job output. The syntax of the **query** command for LDAP is:

```
f  ldapsrv,appl=query,report[,report...][,reset][,console|noconsole]
```

Where *report* is the name of an LDAP performance report, shown below.

reset Used to reset the statistics shown in the various reports.

console|noconsole
console is the default. It indicates the report output should go to the system console as well as the job log. **noconsole** indicates that the report output should only go to the job log.

The following is a list of all the available LDAP performance reports:

ALL This option shows all of the reports described below.

AIO This report shows the TCP/IP asynchronous socket calls made by the LDAP server and the amount of queuing for incoming requests. Following is sample output if you request an **AIO** report:

```
     Asynchronous I/O Statistics
     ---------------------------
Service Thread Stack Size=76K   Number of Pools=3
Pool:   0   Threads:    10     Pool:   1   Threads:    10
Pool:   2   Threads:    10

       SRB Accepts :        100          SRB Requests :    2167242
        SRB Queued :     537203           Pct. Queued :       24.8%
         Schedules :          2               Accepts :        101
             Reads :          0                Readvs :          0
             Recvs :    2167297             Recvfroms :          0
            Writes :          0               Writevs :          0
             Sends :          0               Sendtos :          0
           Cancels :          0         Cancelsockets :          0
```

DEBUG This option shows the current debug setting. Following is sample output if you request a **DEBUG** report:

```
           Current Debug Settings
           ----------------------
debug level: 0 (decimal)
           : 0x0
           : OFF

in-storage trace settings:
--------------------------

osi:         Print Flag=0x00   Threshold=30
```

```
lock:        Print Flag=0x00   Threshold=30
storage:     Print Flag=0x00   Threshold=30
all:         Print Flag=0x00   Threshold=30
asyncio:     Print Flag=0x00   Threshold=30
ldapdebug:   Print Flag=0x00   Threshold=30
```

LOCKING This report shows the lock facility statistics. It includes the number of lock waits, the average lock wait time, and the time threads sleep waiting for certain specific events. Following is sample output if you request a **LOCKING** report:

```
    Locking Statistics
    ------------------

Untimed sleeps:        0  Timed Sleeps:        1  Wakeups:        0

Total waits for locks:                  125352
Average lock wait time:                      0.164 (msecs)

Total monitored sleeps:                      0
Average monitored sleep time:                0.000 (msecs)

                Top 15 Most Highly Contended Locks
  Thread      Async       Spin
   Wait       Disp.       Resol.     Pct.   Description
----------  ----------  ----------  ------  --------------
    100882           0           0   80.4%  DN cache locks
     21287           0           0   17.0%  FILTER cache locks
      1555           0           0    1.2%  ACL SRCMAP cache locks
       926           0           0    0.7%  Cache Manager locks
       497           0           0    0.4%  ACL ACLSRC cache locks
       166           0           0    0.1%  ENTRY cache locks
         0         137           0    0.1%  Global process lock
        42           0           0    0.0%  Unclassified locks
         0          28           0    0.0%  AS queue lock
         0          16           0    0.0%  Global queue of threads waiting for locks
         1           0           0    0.0%  ODBC chain locks
         0           0           0    0.0%  ssl layer locks
         0           0           0    0.0%  record wrapper locks
         0           0           0    0.0%  main layer locks
         0           0           0    0.0%  driver interface locks

Total lock contention of all kinds:     125537
```

MONITOR This report lists various server statistics related to client connections, client requests, and server caching activity.

Following is sample output if you request a **MONITOR** report:

```
     Monitor Statistics
     ------------------

Server Version:      z/OS Version 1 Release 6 Integrated Security Services LDAP Server
Current Time:        Fri Mar 12 21:39:12 2004
Start Time:          Fri Mar 12 18:31:17 2004
Last Reset Time:     Fri Mar 12 18:31:17 2004
Number of Resets:    0

Server Totals:
--------------
Description                 Count
--------------------------------
Config Max Connections       1000
System Max Connections        500
Total Connections             100
Current Connections            31
MaxReached Connections         32
Search Entries Sent        491661
Message Bytes Sent      189547603
```

```
Search References Sent          0

Operation    Requested  Completed
-----------  ---------  ---------
AllOps         2167422    2167422
Abandons             0          0
Adds              5726       5726
Binds              100        100
Compares        279264     279264
Deletes           5716       5716
ExtOps               0          0
Modifies         12614      12614
ModifyDNs            0          0
Searches       1863933    1863933
Unbinds             69         69
Unknown              0          0

Backend Statistics: tdbm1
-----------------
Naming Context:     O=IBM.COM
Naming Context:     SECAUTHORITY=DEFAULT
Naming Context:     CN=EXTRA_SUFFIX

Description                Count
---------------------------------
Search Entries Sent        491518
Message Bytes Sent      189349324
Search References Sent          0

Operation    Requested  Completed
-----------  ---------  ---------
AllOps         2167203    2167203
Abandons             0          0
Adds              5726       5726
Binds                0          0
Compares        279264     279264
Deletes           5716       5716
ExtOps               0          0
Modifies         12614      12614
ModifyDNs            0          0
Searches       1863883    1863883
Unbinds              0          0
Unknown              0          0

                                                    Pct  Refresh  Refresh
Cache Name     Size  Entries     Hits   Misses    Hit    Count  AvgEnts
----------  -------  -------  -------  -------  -----  -------  -------
ACLSource       100      100   465270   280626   62.4%        0        0
Dn             1000      998    35746k   25057k  58.8%        0        0
DnToEID        1000      181  1880880    14714   99.2%        0        0
Entry          5000     4979    81598    21430   79.2%        0        0
EntryOwner      100        2   799028        3  100.0%        0        0
Filter         5000        0   483586  1375339   26.0%   22597        3

Filter Bypass Limit: 100

Backend Statistics: cdbm1
-----------------
Naming Context:     CN=MONITOR

Description                Count
---------------------------------
Search Entries Sent           143
Message Bytes Sent         196831
Search References Sent          0

Operation    Requested  Completed
```

```
-----------  ---------  ---------
AllOps              49         49
Abandons             0          0
Adds                 0          0
Binds                0          0
Compares             0          0
Deletes              0          0
ExtOps               0          0
Modifies             0          0
ModifyDNs            0          0
Searches            49         49
Unbinds              0          0
Unknown              0          0
```

Note: Some statistics may exceed the width of the report columns above. In this case, a suffix is appended indicating the number is expressed in larger units. These suffixes are as follows:

```
k - kilo - displayed number times 10^3
M - mega - displayed number times 10^6
G - giga - displayed number times 10^9
T - tera - displayed number times 10^12
P - peta - displayed number times 10^15
```

When monitor statistics are reset with the **reset** option, the resetting action will be reflected in future MODIFY QUERY reports and *cn=monitor* searches. This resetting action will have no effect on the values reported in the Activity Log. When statistics are reset:

- **Last Reset Time** is set to **Current Time**, **Number of Resets** is incremented, and **MaxReached Connections** is set to the value of **Current Connections**.
- **Search Entries Sent, Message Bytes Sent, Search References Sent**, and all Operation **Requested** and **Completed** values are reset to zero for both the **Server Totals** and **Backend Statistics**.
- Cache statistics **Hits, Misses, Pct Hit, Refresh Count**, and **Refresh AvgEnts** for each cache are reset to zero.

Resetting the statistics has no effect on the Cache **Size** for each cache, nor on the **Filter Bypass Limit**, since these are configured values. Resetting the statistics also has no effect on the Cache **Entries** for each cache, since the contents of the caches are not altered by a reset of statistics.

Some caches may get invalidated and refreshed due to directory update operations. When this occurs, Cache **Refresh Count** is incremented and the Cache **Entries** is set to zero to reflect the refreshed, or empty, cache. The Cache statistics **Hits, Misses**, and **Pct Hit** values are accumulated across cache invalidation and refresh.

THREADS This report lists the state of all the LDAP server threads. Following is sample output if you request a **THREADS** report:

```
            Currently Running Threads
            -------------------------
TCB       Stack     Size      Routine              State
--------  --------  --------  --------             --------
006DA5A8  1D4E0270     77824                       WAITECB   off=E1B881C0  rtn=
006E3160  1D411060     77824                       WAITECB   off=E1B881C0  rtn=
006C38C8  1D59AC48     77824                       WAITECB   off=E1B881C0  rtn=
006C3E88  1D554AB0     77824                       WAITECB   off=E1B881C0  rtn=
006DB730  1D487710     77824                       WAITECB   off=E1B881C0  rtn=
006DBCF0  1D458130     77824                       WAITECB   off=E1B881C0  rtn=
006DCCF0  1D5FA940     77824                       WAITECB   off=E1B881C0  rtn=
006C7E88  1EA086E0     77824                       WAITECB   off=E1B881C0  rtn=
006DA278  1D50D9E0     77824                       WAITECB   off=E1B881C0  rtn=
006E33F0  1D3E29B8     77824                       WAITECB   off=E1B881C0  rtn=
```

```
006DC638  1E9AFBF0     77824                    WAITECB   off=E1B881C0  rtn=
006DA740  1D4F6E28     77824                    WAITECB   off=E1B881C0  rtn=
006C3A60  1D56D4D8     77824                    WAITECB   off=E1B881C0  rtn=
006DBE88  1D427C18     77824                    WAITECB   off=E1B881C0  rtn=
006DC188  1E9F0BF0     77824                    WAITECB   off=E1B881C0  rtn=
006DC8C8  1D612430     77824                    WAITECB   off=E1B881C0  rtn=
006DCE88  1D5CD1D0     77824                    WAITECB   off=E1B881C0  rtn=
006C7A60  1EA486E0     77824                    WAITECB   off=E1B881C0  rtn=
006DA410  1D524598     77824                    WAITECB   off=E1B881C0  rtn=
006DAD90  1D4A0138     77824                    RUNNING
006C33A8  1D5B2738     77824                    WAITECB   off=E1B881C0  rtn=
006DB0D0  1D4B6CF0     77824                    WAITECB   off=E1B881C0  rtn=
006E3588  1D3F9570     77824                    WAITECB   off=E1B881C0  rtn=
006C3CF0  1D53DEF8     77824                    WAITECB   off=E1B881C0  rtn=
006DB598  1D46FC20     77824                    WAITECB   off=E1B881C0  rtn=
006DC418  1E9C76E0     77824                    WAITECB   off=E1B881C0  rtn=
006DCB58  1D5E3D88     77824                    WAITECB   off=E1B881C0  rtn=
006C3638  1D584090     77824                    WAITECB   off=E1B881C0  rtn=
006E3818  1D3DBE70      8192                    CONDWAIT
006DBA60  1D441578     77824                    WAITECB   off=E1B881C0  rtn=
006C7BF8  1EA30BF0     77824                    RUNNING
```

Activity Logging

The LDAP server can record server activity for the purpose of load analysis. The log file can contain information about operations handled by the server, client IP addresses, messages generated by the server, and summary statistics. In general, each record consists of a time stamp followed by the record data. Operations are logged when they successfully begin processing. Optionally, a log record can be created when the operation completes processing. The following is an example of a typical operation log record followed by the ending log record.

```
Wed May 22 11:53:52 2002 Search: connid = 4, base = cn=monitor, filter = (objectclass=*), IP = 1.2.3.4
Wed May 22 11:53:52 2002 End Search: connid = 4, base = cn=monitor, filter = (objectclass=*),
rc = 0, IP = 1.2.3.4
```

Note that when the client is connecting to the LDAP server over the PC interface, the IP address will be reported as 'PC'.

Log records created for message logging appear in the log file with the timestamp followed by the message. The following is an example of a message log record.

```
Wed May 22 11:30:49 2002 GLD0210I Modify command has been processed successfully: LOG,MSGS
```

Summary records are created on an hourly basis as long as log records are being collected or when a modify command is processed that affects the log function. The summary log records contain information about the operations that the server has processed. The following is an example of the summary log records.

```
Wed May 22 11:45:25 2002 total operations started = 3662306
Wed May 22 11:45:25 2002 total operations completed = 3662304
Wed May 22 11:45:25 2002 total abandons completed = 0
Wed May 22 11:45:25 2002 total adds completed = 0
Wed May 22 11:45:25 2002 total binds completed = 1245
Wed May 22 11:45:25 2002 total compares completed = 0
Wed May 22 11:45:25 2002 total deletes completed = 0
Wed May 22 11:45:25 2002 total extendedops completed = 0
Wed May 22 11:45:25 2002 total modifies completed = 47
Wed May 22 11:45:25 2002 total modifydns completed = 0
Wed May 22 11:45:25 2002 total searches completed = 3661012
Wed May 22 11:45:25 2002 total unbinds completed = 0
Wed May 22 11:45:25 2002 total unknown ops completed = 0
Wed May 22 11:45:25 2002 total search entries sent = 4123452
Wed May 22 11:45:25 2002 total search references sent = 0
Wed May 22 11:45:25 2002 total bytes sent = 572465214
```

```
| Wed May 22 11:45:25 2002 total connections processed = 2
| Wed May 22 11:45:25 2002 current connections = 2
| Wed May 22 11:45:25 2002 connection high water mark = 2
```

The location of the log file is controlled by the **logfile** configuration file option. The activity log can be written to either an HFS file or a MVS dataset. If the log file is an MVS dataset, it must be created (allocated) prior to its use by the LDAP server. It is recommended that when an HFS file is desired that the file specification be fully qualified. The following is an example of a logfile option specifying an HFS file.

```
logfile /etc/ldap/gldlog.output
```

`/etc/ldap/gldlog.output` is also the default location of the log file.

The MVS dataset can be specified through either a ddname or as a specific dataset. The following is an example for both methods.

```
logfile //dd:logout
logfile //'mysys.ldap.actlog'
```

The default data collection setting is to collect no data. The default is modified by either environment variables or through the specification of operator modify commands. The environment variables are read as the server starts up while the modify commands can be specified once the server has started.

A modify command can be sent to the LDAP server from the SDSF or the operator's console. Note that if the command is entered form SDSF, it must be preceded by a slash (/). In the command,

```
f ldapsrv,appl=log,setting
```

the *setting* is the modification to the log collection. To send the same command to the LDAP server in the z/OS shell, it is necessary to know the job name assigned to the process. Do this by performing

```
d a,l
```

from SDSF or the operator's console and determine the name which includes the user ID under which the LDAP server is running and a suffix. Once this name is found, use it to replace ldapsrv in the command above.

The **GLDLOG_OPS** environment variable or the modify command controls which operations generate log entries. Prior to specifying an operation setting, no operation logging is performed. The settings are:

- **writeops** indicates that log entries are to be created at the start of add, delete, modify, modrdn and extended operations.
- **allops** indicates that log entries are to be created as for writeops and in addition, search and compare operations are to be logged.
- **summary** indicates that only hourly summary statistics are to be logged. As long as any logging is being collect, summary data is produced on an hourly basis.

The **GLDLOG_TIME** environment variable or the modify command controls whether log entries are generated when logged operation ends. The settings are:

- **time** indicates that ending logs are to be created for all operations that are being logged.
- **notime** indicates that ending logs are not to be created for operations that are being logged. This is the default prior to specifying the **GLDLOG_TIME** environment variable or by issuing the modify command to set the operation end time control.

The **GLDLOG_MSG** environment variable or the modify command controls whether log entries are generated when messages are created by the LDAP server. The settings are:

- **msgs** indicates that messages generated by the LDAP server are to be written to the activity log in addition to the normal target.

- **nomsgs** indicates that messages generated by the LDAP server are not to be written to the activity log. This is the default prior to specifying the GLDLOG_MSG environment variable or by issuing the modify command to set the message control.

As the log entries are produced some buffering of the output is performed by the system. The buffers are flushed before the server shuts down. However, you can force the server to flush the buffers by entering a modify command:

```
f ldapsrv,appl=log,flush
```

The server can be told to stop collecting activity data by entering a modify command:

```
f ldapsrv,appl=log,stop
```

Monitoring client connections

As the number of concurrent client connections approaches the maximum number of client connections allowed on the LDAP server, the LDAP Server issues warning messages to the console when additional clients attempt to bind to the LDAP server. To avoid overloading the console with messages, these warning messages are issued, at most, once per minute for a maximum of 60 times while the number of concurrent client connections remains at a high level. If the number of concurrent client connections on the LDAP server falls below a safe threshold, another console message is issued stating that the number of concurrent client connections is now at a safe level. After this, the cycle of warning messages can begin again if the number of concurrent client connections again approaches the maximum number of connections allowed on the LDAP server.

The issuance of these console warning messages on a fairly regular basis may signify that the **maxConnections** option in the LDAP configuration file is set to a low value and should be increased. It is also possible that the MAXFILEPROC statement or the MAXSOCKETS option on the NETWORK statement within BPXPRMxx may need to be adjusted upward to support a higher value of **maxConnections**. The activity log on the LDAP Server can be used to monitor the number of client connections. Refer to “Activity Logging” on page 113 for more information on activity logging.

Using the LDAP server for PC callable support

When the LDAP server comes up, it tries to enable any interfaces that are configured to it. If at least one interface comes up, the LDAP server remains up. If you are using the program call (PC) support in the LDAP server to provide Policy Director access to LDAP data, it is recommended that you use the LDAP server only for that type of interface and no other interfaces. At most one LDAP server in a system can activate PC callable support. If an LDAP server tries to initialize PC callable support after another LDAP server has attempted (successfully or unsuccessfully) to initialize PC callable support, the initialization fails. The previous server that has locked access to the PC callable support must be shut down before another server can attempt to run with the PC callable interface.

Chapter 10. Migrating to a z/OS LDAP server

This chapter discusses migration issues. Your plan for migrating to a z/OS LDAP server should include information from a variety of sources. These sources of information describe topics such as coexistence service and optional migration actions. See the *z/OS Migration* , on how to migrate from various z/OS releases. Also, see the *z/OS Summary of Message and Interface Changes*, for interface changes.

The following documentation, which is supplied with your product order, provides information about installing your z/OS system. In addition to specific information about the LDAP server, this documentation contains information about all of the z/OS elements.

- *z/OS and z/OS.e Planning for Installation*

  This book describes the installation requirements for z/OS at a system and element level. It includes hardware, software, and service requirements for both the driving and target systems. It also describes any coexistence considerations and actions.
- *z/OS Program Directory*

  This document, which is provided with your z/OS product order, leads you through the specific installation steps for the LDAP server and the other z/OS elements.
- *ServerPac: Installing Your Order*

  This is the order-customized, installation book for using the ServerPac Installation method. Be sure to review the /usr/lpp/ldap/ldap/etc/slapd.conf file, which describes data sets supplied, jobs or procedures that have been completed for you, and product status. IBM may have run jobs or made updates to PARMLIB or other system control data sets. These updates could affect your migration.

Within this chapter, you can find information about the specific updates and considerations that apply to this release of z/OS LDAP.

- "Migration roadmap" on page 122

  This section identifies the migration paths that are supported with the current level of the LDAP server. It also describes the additional publications that can assist you with your migration to the current level.
- "z/OS V1R6 overview" on page 123

  This section describes the specific updates that were made to LDAP for the current release. For each item, this section provides an overview of the change, a description of any migration and coexistence tasks that may be considered, and where you can find more detailed information in the z/OS LDAP library or other element libraries.

Actions required for migrations

The following sections describe common activities and considerations that are typically required (or should be considered) whenever you migrate from a previous release of the LDAP server to the current release of the LDAP server.

After each heading in this section, a table indicates the z/OS LDAP releases you could be migrating from, showing an **X** for each release the particular section pertains to. If, for example, you are migrating from z/OS V1R4, the sections that show an **X** under z/OS V1R4 and z/OS V1R5 apply to your installation and should be considered.

Obtaining the DB_VERSION setting for RDBM

z/OS V1R3	z/OS V1R4 and z/OS V1R5
X	

Some of the steps in the following migration actions depend on the **DB_VERSION** setting for existing RDBM DB2 tables. The **DB_VERSION** can be obtained using the following SQL operation entered through SPUFI:

```
SELECT DB_VERSION FROM USERID.LDAP_NEXT_EID
```

Where *USERID* is the value of the **dbuserid** option in the configuration file.

Obtaining the DB_VERSION setting for TDBM

z/OS V1R3	z/OS V1R4 and z/OS V1R5
X	X

Some of the steps in the following migration actions depend on the **DB_VERSION** setting for existing TDBM DB2 tables. The **DB_VERSION** can be obtained using the following SQL operation entered through SPUFI:

```
SELECT DB_VERSION FROM USERID.DIR_MISC
```

Where *USERID* is the value of the **dbuserid** option in the configuration file.

Migrating to new dataset name for LDAP executable code

z/OS V1R3	z/OS V1R4 and z/OS V1R5
X	X

The LDAP executable code is now installed in SYS1.SIEALNKE, instead of *GLDHLQ*.SGLDLNK. By default, SYS1.SIEALNKE is APF authorized and is in the LINKLIST (thus does not need to be specified on a STEPLIB). Review your usage of the LDAP executable dataset name and replace the name or remove the usage as appropriate.

Note: The names of the other LDAP datasets have not changed: *GLDHLQ*.SGLDSAMP, *GLDHLQ*.SGLDEXEC, *GLDHLQ*.SGLDHDRC, and *GLDHLQ*.SGLDEXPC.

Schema migration

z/OS V1R3	z/OS V1R4 and z/OS V1R5
X	X

The TDBM schema has been updated. The application of the schema shipped in z/OS V1R6 is recommended after starting the z/OS V1R6 LDAP server. See “Updates to the schema” on page 166 for additional information. All future schema service will assume that these updates have been applied to customer schemas.

TDBM schema migration

z/OS V1R3	z/OS V1R4 and z/OS V1R5
X	X

To migrate the TDBM schema from a z/OS V1R5 or earlier LDAP server to a z/OS V1R6 LDAP server:

1. Check that **schemaReplaceByValue off** is not specified in the z/OS V1R6 LDAP configuration file. Start the z/OS V1R6 LDAP server with your current TDBM backend.

2. If the **schema.user.ldif** file was used to create the schema in use by TDBM
   a. Copy /usr/lpp/ldap/etc/schema.user.ldif to /etc/ldap. Edit the file and replace <suffix> with one of the suffixes in your TDBM backend.
   b. Issue the following ldapmodify command:

      ```
      ldapmodify -h host -p port -D adminDN -w adminPW -f /etc/ldap/schema.user.ldif
      ```

3. If the **schema.IBM.ldif** file was used to create the schema in use by TDBM
   a. Copy /usr/lpp/ldap/etc/schema.IBM.ldif to /etc/ldap. Edit the file and replace <suffix> with one of the suffixes in your TDBM backend.
   b. Issue the following ldapmodify command:

      ```
      ldapmodify -h host -p port -D adminDN -w adminPW -f /etc/ldap/schema.IBM.ldif
      ```

Migrating TDBM databases

z/OS V1R3	z/OS V1R4 and z/OS V1R5
X	X

To use the new support for expanded static, dynamic, and nested groups added in z/OS V1R6, the TDBM database version (as defined in the DB2 tables allocated for the TDBM database backend) must be set to 3.0. For database instances created using z/OS V1R6, the database version will be set to 3.0 as part of first starting up either the LDAP server or the **ldif2tdbm** programs. For database instances created using an earlier release, the TDBM database version must be updated in order to use the enhanced group support.

If you are running the LDAP server in a sysplex environment with mixed levels of systems, you cannot use the enhanced group support until all LDAP servers accessing the same TDBM database instance are running at the z/OS V1R6 level. Once all these systems are running z/OS V1R6, the database version can be set to 3.0 and the expanded static, dynamic, and nested group support can be used.

Similarly, the access control grant, deny, and attribute-level permissions capabilities introduced in z/OS V1R4 require that the TDBM database version be set to 2.0 or 3.0. For database instances created using z/OS V1R3 or earlier, the TDBM database version must be updated in order to use the enhanced access control features. If you are running the LDAP server in a sysplex environment with earlier levels of z/OS, you cannot use the enhanced access control support until all LDAP servers accessing the same TDBM database instance are running z/OS V1R4 or later. Once all systems are running z/OS V1R4 or later, the database version can be set to 2.0 (if one or more of the systems is z/OS V1R4 or z/OS V1R5) or 3.0 (if all of the systems are z/OS V1R6), and the enhanced access control capabilities can be used.

To change the database version of an existing TDBM database instance that was created using an LDAP server from a release prior to z/OS V1R6, run the following SQL using either the SPUFI facility or suitable program (for example, DSNTIAD). Specify '3.0' to set the version for z/OS V1R6 and '2.0' for z/OS V1R4 or z/OS V1R5.

```
UPDATE USERID.DIR_MISC SET DB_VERSION='3.0';
```

where USERID is set to be the value of the **dbuserid** LDAP server configuration file setting for the TDBM backend instance.

If there are enhanced static, dynamic, or nested groups present in the TDBM backend, you can verify that the enhanced group support is enabled by examining the LDAP server or **ldif2tdbm** output log for the either of these two messages:

```
GLD3147I Dynamic, nested, and expanded static group membership determination is available
 but not in use below suffixes: suffix-list.
GLD3146I Dynamic, nested, and expanded static group membership determination is in use
 below suffixes: suffix-list.
```

To verify that the enhanced access control capabilities are enabled, examine the LDAP server or **ldif2tdbm** output log for this message:

```
GLD3135I Grant/Deny ACL support is enabled below suffixes: suffix-list.
```

Fallback procedures in case a prior release of the LDAP server must be run

If it becomes necessary to fall back to a prior release of the LDAP server, it is possible to set the TDBM database version back to '2.0'. By doing this, enhanced group capabilities will be turned off and any expanded static, nested, or dynamic groups that were set up while the TDBM database version was set to '3.0' will be ignored for group gathering and access control checking.

Note: When running in a fallback mode, the only group definitions that are evaluated are static groups that have an object class of **accessGroup**.

Similarly, the TDBM database version can be set back to '1.0'. In addition to disabling the enhanced group capabilities as described above, this will also turn off the enhanced access control capabilities.

Note: Any enhanced access controls that were previously set up will be ignored for access control checking. Any entries which contain enhanced access controls will not be modifiable while running in this fall back mode.

To set the TDBM database version back, run the following SQL, using '2.0' to reset to z/OS V1R4 and z/OS V1R5 level, and '1.0' to reset to z/OS V1R3 level:

```
UPDATE USERID.DIR_MISC SET DB_VERSION='2.0';
```

If there are enhanced static, dynamic, or nested groups present in the TDBM backend, you can verify that the enhanced group support is disabled by examining the LDAP server or **ldif2tdbm** output log for this message:

```
GLD3148I Dynamic, nested, or expanded static group data is present in the TDBM
 backend but ignored since the DB_VERSION is not 3.0 or greater below
 suffixes: suffix-list.
```

To verify that the enhanced access control capabilities are disabled, examine the LDAP server or **ldif2tdbm** program output log for this message:

```
GLD3136I Grant/Deny ACL support is not enabled below suffixes: suffix-list.
```

Coexistence and migration with previous releases (RDBM)

RDBM has been removed. All customers using RDBM must migrate to TDBM.

Because TDBM has a different DB2 table design than RDBM, DB2 data sharing between TDBM and existing RDBM tables is not possible. This means that existing RDBM data must be unloaded and then loaded into TDBM.

Note: You must unload the RDBM data before installing the new release of the LDAP server because RDBM is not supported in the new release.

Migrating existing RDBM data to a TDBM database

z/OS V1R3	z/OS V1R4 and z/OS V1R5
X	

1. Create a backup of the existing RDBM directory data using native DB2 utilities.
2. The RDBM database must be unloaded on the existing release **prior to moving to the new release**. Unload the RDBM directory information using **db2ldif**. As an example:

   ```
   db2ldif -f /etc/ldap/slapd.conf -o /tmp/directorydata.unload
   ```

assuming this command was run from the z/OS shell prompt.

3. Obtain the **DB_VERSION** setting for the existing RDBM DB2 tables (see “Obtaining the DB_VERSION setting for RDBM” on page 117 for more information). If the **DB_VERSION** is 8.0, skip to step 4. Otherwise, evaluate all comments (lines beginning with a pound sign (#)) that begin with AF, AP, DF, or DP in the LDIF file created in the previous step. If the attribute value or distinguished name described in this comment is acceptable to you, no further updates are necessary for that attribute value. If the attribute value is not acceptable to you, modify the uncommented LDIF line below the comments to contain an attribute value acceptable to you.
4. Create the TDBM database and table spaces in DB2. See “Creating the DB2 database and table spaces for TDBM or GDBM” on page 42 for these instructions.
5. Make appropriate configuration changes. See “Configuration file checklist” on page 54 for information on the TDBM configuration options. The RDBM section and the includes for schema files should be removed.
6. The schema definitions associated with the TDBM backend conform to the LDAP Version 3 (V3) protocol. The schema files that are shipped with z/OS LDAP to be used with TDBM are a superset of the schema definitions shipped for use with RDBM. The schema files shipped for TDBM are in the LDAP V3 protocol format while the schema files shipped for RDBM were in LDAP V2 protocol format. If you have modified the LDAP V2 protocol schema files used with RDBM, corresponding changes may need to be made to the LDAP V3 protocol format schema files used with TDBM. If you have added schema definitions, then corresponding schema definitions may need to be added to the LDAP V3 protocol schema files. See “Migrating the schema from RDBM to TDBM” on page 183 for more information.
7. Load the data from the LDIF file just created using either the **ldif2tdbm** utility or the **ldapadd** utility. See “ldif2tdbm utility” on page 138 for details, including instructions on determining which utility is best suited for your installation.
8. Optionally, remove the old DB2 tables. To do this, remove the tablespaces for the RDBM database. The following command should be repeated for each tablespace associated with the RDBM database. Examine the SPUFI used to create the RDBM database to determine the tablespace names.

   ```
   DROP TABLESPACE <tbspname>
   ```

   This step is optional because RDBM and TDBM databases can coexist.

Migrating RDBM to TDBM: entryOwner and aclEntry attribute value changes

z/OS V1R3	z/OS V1R4 and z/OS V1R5
X	

With RDBM, it is possible to add **entryOwner** and **aclEntry** attribute types where the distinguished name (DN) portion of the attribute value contains an incorrectly formatted DN. For example, unescaped special characters (such as the # sign) are allowed. TDBM does not support incorrectly formatted DNs in **aclEntry** and **entryOwner** attribute values. Attribute values of this type containing correctly formatted DNs can be migrated to TDBM without change. Attribute values of this type containing incorrectly formatted DNs must be manually corrected prior to being loaded into TDBM. To date, this problem has only been encountered with SDBM DNs present in RDBM **aclentry** and **entryOwner** values that were being migrated to TDBM. See IETF RFC 2253, *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names* for rules that define a correctly formatted DN.

Migrating RDBM to TDBM: timestamp changes

z/OS V1R3	z/OS V1R4 and z/OS V1R5
X	

If you are migrating from an RDBM backend to a TDBM backend, the timestamps are different. Timestamp attribute values generated by the RDBM backend (**modifytimestamp** and **createtimestamp**) were of the following format:

```
modifytimestamp=2000-05-05 10:30:56.128806
```

This value represents the ″local time″. In this example, the local time is with respect to eastern standard time timezone. (Note that ″daylight savings time″ is in effect.)

Timestamp attribute values generated by TDBM are in the Generalized Time format with respect to GMT time. In TDBM, the previously listed timestamp would be:

```
modifytimestamp=20000505143056.128806Z
```

Removing schema file include statements

z/OS V1R3	z/OS V1R4 and V1R5
X	

Remove the **include** statements in your configuration file for any files that contain **attribute** or **objectclass** lines. These definitions were needed for SDBM and RDBM in releases prior to z/OS V1R4. The **attribute** and **objectclass** configuration options are no longer supported. If you have defined your own attributes and objectclasses for use with RDBM, refer to “Schema migration” on page 118 for information about converting your definitions for use with TDBM.

Removing unused configuration option

z/OS V1R3	z/OS V1R4 and V1R5
X	

The following configuration options are no longer necessary or evaluated and should be removed from the configuration file: **verifySchema, attribute, objectclass, replKeyRingFile, replKeyRingPW, rdbm dbtype** on the **database** option, **index, tbspaceentry, tbspacemutex, tbspace32k, tbspace4k**. See the description of **sslKeyRingFile** in Chapter 8, “Customizing the LDAP server configuration,” on page 53 for information on using key rings.

Migration roadmap

This section describes the migration paths that are supported by the current release of the LDAP server. It also provides information about how you can obtain the LDAP server migration information from previous releases.

z/OS V1R6 update summary

The following table summarizes the updates introduced to the z/OS LDAP server in z/OS Version 1 Release 6 (z/OS V1R6). If you are migrating from z/OS V1R4, z/OS V1R3, z/OS V1R2, z/OS V1R1, or OS/390® V2R10, you should review the information in the detailed section for each item.

Table 19. Summary of LDAP server updates for z/OS V1R6

For Information About:	Refer to Page:
Expanded static, dynamic, and nested groups	257
Alias support	301

Table 19. Summary of LDAP server updates for z/OS V1R6 (continued)

For Information About:	Refer to Page:
Change logging support for TDBM	307
Enhanced schema update	165
DB2 restart/recovery	106
Enhanced monitor support	109
Entry and filter cache support	355

z/OS V1R5 update summary

No new function was introduced to the z/OS LDAP server for this release.

z/OS V1R4 update summary

The following table summarizes the updates introduced to the z/OS LDAP server in z/OS Version 1 Release 4 (z/OS V1R4). If you are migrating from z/OS V1R3, z/OS V1R2, z/OS V1R1, or OS/390 V2R10, you should review the information in the detailed section for each item.

Table 20. Summary of LDAP server updates for z/OS V1 R4

For Information About:	Refer to Page:
Modify DN	191
CRAM-MD5 and DIGEST-MD5 authentication	251
Transport Layer Security (TLS) support	46
ACL enhancements	269
SDBM enhancements	41
LDAP IBM-entryuuid support	135
Activity logging	101
Abandon support	335
RDBM removal	120
Monitor support	335

z/OS V1R3 update summary

No new function was introduced to the z/OS LDAP server for this release.

z/OS V1R6 overview

This section describes the new and changed LDAP server functions introduced for z/OS Version 1 Release 6 (V1R6). The information about each item includes:

- Description
- Summary of the LDAP server tasks or interfaces that may be affected
- Coexistence considerations, if any, that are associated with the item
- Migration procedures, if any, that are associated with the item
- References to other publications that contain additional detailed information.

Expanded static, dynamic and nested groups

Description

Additional group definitions have been added to the TDBM backend. The TDBM backend now supports expanded static, dynamic, and nested group definitions. Static group entries can now have one or more of the following objectclasses:

- **accessGroup**
- **accessRole**
- **ibm-staticGroup**
- **groupOfNames**
- **groupOfUniqueNames**

Dynamic group entries have one of the following objectclasses:

- **groupOfUrls**
- **ibm-dynamicGroup**

Nested group entries are defined with the objectclass **ibm-nestedGroup**. These new group definitions can now be used for group gathering at authentication time. Also, these new group definitions are supported by the **ibm-allMembers** and **ibm-allGroups** search and comparison operations.

What this change affects

This change affects the groups that are gathered during authentication time. In previous z/OS LDAP releases, only static groups with an objectclass of **accessGroup** were gathered during authentication time.

Dependencies

None.

Coexistence considerations

To properly use this support, all LDAP servers that are using the same set of DB2 tables (TDBM backend database instance) should be capable of interpreting the new group definitions that are now supported to properly obtain all groups that a binding user belongs to during authentication. The new group definitions can also affect the results of **ibm-allGroups** and **ibm-allMembers** search and comparison operations. When running in a sysplex or multi-server mode, all LDAP servers must be at the z/OS V1R6 level before this support can be enabled. This support is enabled by modifying the DB_VERSION value for the TDBM backend database instance. New TDBM backend database instances, created using the z/OS V1R6 LDAP server, will initialize the DB_VERSION value to '3.0' to allow the new group definitions to be used in the newly defined TDBM backend database instance. In order to use such a TDBM backend database in a sysplex or multi-server mode with the z/OS V1R4 LDAP server, the DB_VERSION must be set to '2.0'. In order to use such a TDBM backend database in a sysplex or multi-server mod with the z/OS V1R3 LDAP server, the DB_VERSION must be set to '1.0'.

Migration tasks

In order to use the new group definitions that are provided in z/OS V1R6, the TDBM database version (as defined in the DB2 tables allocated for the TDBM database backend) must be set appropriately. See "Migrating TDBM databases" on page 119 for information on how to do this.

For more information

See Chapter 21, "Static, dynamic, and nested groups," on page 257 for more information on expanded groups.

Alias support

Description

Alias support provides a means for a TDBM directory entry to point to another entry in the same TDBM directory. If a distinguished name encountered during a search operation with dereferencing contains an alias, the alias is replaced by the value it points to and search continues using the new distinguished name.

An alias can be used to create a convenient public name for an entry or subtree, hiding the more complex actual name of the entry or subtree. Aliasing is only supported in the TDBM (DB2 based) backend.

What this change affects

Search operations with dereferencing now respect alias entries. In previous releases, dereferencing was ignored and alias entries were always processed as normal entries.

Usage of aliases in a directory can cause a large increase in the amount of processing that takes place during search, even if no alias entries are actually involved in the particular search that was requested. See Chapter 24, “Alias,” on page 301 for information on how to minimize the impact to search performance.

Dependencies

None.

Coexistence considerations

Do not use aliasing in a sysplex with different levels of LDAP server that are using the same set of DB2 tables. Alias entries created on a previous level of LDAP server may not be valid, leading to search errors when searching with dereferencing. Also, since alias dereferencing is not supported in earlier releases, a search with dereferencing can return different results than when performed on z/OS V1R6 or later release.

Migration tasks

If an existing TDBM directory already contains alias entries, you must ensure that:

- each alias entry is a leaf entry (has no children)
- the **aliasedObjectName** attribute in the alias entry does not point to the alias entry, to a descendant of the alias entry, or to the schema entry
- the alias entry is not also a referral

Also, the TDBM schema entry cannot be an alias.

Dereferencing during search is controlled by a flag set by the LDAP client on a search request. In some clients, such as the z/OS LDAP client, the default value for this flag is to do no dereferencing, thus returning the same results as the search on a previous release of z/OS. Other clients may have a different default. In particular, the default for the JNDI client is to always dereference during search. Check that your applications will continue to work correctly with the default dereferencing flag value for your LDAP client. If not, you will need to change your applications to override the default setting.

For more information

See Chapter 24, “Alias,” on page 301 for more information on using aliases.

Change logging support for TDBM

Description

The change logging support added to earlier releases provided a new LDAP directory in which RACF could create entries containing information about changes to a RACF user. The change logging support is now extended to TDBM backends, allowing a TDBM backend to create entries in the same change log for changes to directory entries in the TDBM backend. In addition, there is limited support for also logging changes to the change log itself (which is contained in the GDBM backend).

What this change affects

Add, modify, delete, and rename operations on entries in a TDBM backend can result in the creation of a change log entry.

Dependencies

None.

Coexistence considerations

In a sysplex with different levels of LDAP server that are using the same set of DB2 tables, change log records are only created for changes to a TDBM entry when that change is made via a z/OS V1R6 or later LDAP server.

Migration tasks

By default, change logging is performed for changes to an entry in a TDBM backend. Thus, if your existing LDAP server has change logging configured (for RACF user changes) and also contains a TDBM backend, then change log records will be created for changes to entries in the TDBM backend when you migrate to z/OS V1R6 or later release. If you do not want to log changes to the TDBM entries, then add **changeLoggingParticipant no** to the TDBM backend section of the configuration file.

For more information

See Chapter 25, “Change logging,” on page 307 for more information on logging changes for TDBM.

Enhanced schema update

Description

When modifying the TDBM or GDBM schema, the modify operation has been enhanced to allow:

- replacing individual values in the schema without affecting other existing values
- changing the Numeric Object Identifier (NOID) of a value in the schema

This simplifies updating the TDBM schema.

What this change affects

The behavior of a modify operation of the schema can be altered.

Dependencies

None.

Coexistence considerations

In a sysplex with different levels of LDAP server that are using the same set of DB2 tables, a modify operation of the schema can behave differently depending on the level of the LDAP server where the operation is processed.

Migration tasks

By default, a modify operation with replace values for the TDBM or GDBM schema will use the new schema-replace-by-value behavior. If this is not desired, either add **schemaReplaceByValue off** to the TDBM or GDBM backend section of the configuration file, or include the **schemaReplaceByValueControl** on the modify request.

For more information

See Chapter 14, “LDAP directory schema,” on page 165 for more information on updating the schema.

DB2 restart/recovery

Description

When using the TDBM or GDBM backend, the LDAP server detects when DB2 shuts down. Based on the **db2terminate** configuration option, the LDAP server can either shut down or continue running and reconnect to DB2 when DB2 is restarted.

What this change affects

This function affects the availability of the LDAP server.

Dependencies

None.

Coexistence considerations

In a sysplex with different levels of LDAP server that are using the same set of DB2 tables, only the z/OS V1R6 LDAP servers will monitor DB2 and handle a DB2 shut down. In a sysplex, it is recommended that the LDAP server shut down so that LDAP requests to the sysplex will be routed to other LDAP servers which still have access to DB2.

Migration tasks

By default, the LDAP server will continue running when DB2 shuts down. If this is not desired, add **db2terminate terminate** to the global section of the configuration file.

For more information

See “DB2 and TCP/IP termination” on page 106 for more information on handling a DB2 shut down.

Enhanced monitor support

Description

The LDAP server now keeps detailed statistics on the activity of each backend, in addition to more information on the LDAP server activity. These statistics can be viewed using an LDAP search operation with a base of ’cn=monitor’ or by a console modify command. When using LDAP search, the statistics for each backend are returned in a separate entry.

What this change affects

The information returned by the monitor.

Dependencies

None.

Coexistence considerations

None.

Migration tasks

Applications which use the monitor statistics may need to be updated to handle the additional entries and information that can be retrieved.

For more information

See “Capturing performance information” on page 109 for more information on the enhanced monitor support.

Entry and filter cache support

Description

The TDBM backend supports caching search results. Entries returned by a search can be stored in the entry cache. Information pertaining to a search (for example, its base, scope, filter, and list of returned entries) can be saved in the filter cache. When a search request is received, TDBM will try to use the filter cache and entry cache to determine the search results and return the matching entries to the caller. The caches are emptied every time an update is made to the TDBM backend.

Entry and filter caching is also available in GDBM, but will not be effective because the change log is more volatile.

What this change affects

Search performance can be improved for a TDBM backend where the same search is issued multiple times.

Dependencies

None.

Coexistence considerations

Entry and filter caching is not supported in multi-server mode (including in a sysplex).

Migration tasks

None.

For more information

See Chapter 29, “Performance tuning,” on page 355 for more information on the entry and filter caches.

z/OS V1R4 overview

This section describes the new and changed LDAP server functions introduced for z/OS Version 1 Release 4 (V1R4). The information about each item includes:

- Description
- Summary of the LDAP server tasks or interfaces that may be affected
- Coexistence considerations, if any, that are associated with the item
- Migration procedures, if any, that are associated with the item
- References to other publications that contain additional detailed information.

Modify DN

Description

Following is a description of each Modify DN enhancement:

- A Modify DN operation can now be accompanied by the new superior parameter. This parameter specifies the DN that will become the immediate superior of the renamed entry.
- A non-leaf node can now be the target of a Modify DN operation.
- A subtree move can be accomplished by specifying a New Superior parameter when the target of the Modify DN operations is a non-leaf node.
- The new **IBMModifyDNRealignDNAttributesControl** can accompany a Modify DN operation. This control will cause the server to search for all attributes of DN syntax whose values match the old DN value. These values will then be updated to reflect the new DN that was created in the Modify DN operation.

What this change affects

Modify DN operations will now affect Replication. For more information see Chapter 15, “Modify DN Operations,” on page 191.

Dependencies

None.

Coexistence considerations

Replication of enhanced Modify DN operations only works with z/OS V1R4 and above LDAP servers. For more information see Chapter 15, “Modify DN Operations,” on page 191.

Migration tasks

None.

For more information

See Chapter 15, “Modify DN Operations,” on page 191 for more information on modify DN.

CRAM-MD5 and DIGEST-MD5 authentication

Description

The z/OS LDAP server has been enhanced to allow CRAM-MD5 and DIGEST-MD5 authentication.

What this change affects
Allows SASL bind mechanisms of CRAM-MD5 and DIGEST-MD5.

Dependencies
None.

Coexistence considerations
None.

Migration tasks
None.

For more information
See Chapter 19, “CRAM-MD5 and DIGEST-MD5 Authentication,” on page 251 for more information on CRAM-MD5 and DIGEST-MD5.

Transport Layer Security (TLS) support

Description
The z/OS LDAP server and client APIs provide support through System SSL for Transport Layer Security (TLS) protocol protection of client/server communication. Transport Layer Security is based upon Secure Sockets Layer (SSL) Security version 3. The differences between TLS v1.0 and SSL v3.0 protocol are not dramatic, but they are significant enough that TLS v1.0 and SSL v3.0 do not inter operate. The z/OS LDAP server and client continues to support SSL v3.0 protocol protection through System SSL. System SSL determines the protocol to use during the secure protocol handshake performed by the peers.

The z/OS LDAP server supports the Start TLS extended operation as described in RFC 2830. This extended operation allows a client communicating over a non-secure connection to change communication to secure communication protected by SSL/TLS. The client may later terminate secure communication while maintaining the connection to the server resulting in non-secure communication between the server and the client.

What this change affects
This change allows the z/OS LDAP server and client APIs to use Transport Layer Security for secure communication. In addition, the LDAP server supports the Start TLS extended operation to change a non-secure connection to a secure connection. The z/OS client APIs do not support the Start TLS extended operations.

Dependencies
z/OS V1R4 or later level of System SSL is required.

Coexistence considerations
None.

Migration tasks
None.

For more information
See “Setting up for SSL/TLS” on page 46 on TLS support.

ACL enhancements

Description
Additional capabilities have been added to the TDBM backend database to support grant/deny as well as attribute-level access control permissions. Access control lists (ACLs) can now contain permission clauses which explicitly deny access to information. In prior releases, access could only be granted, with access to

all other information being denied implicitly. ACLs can now be set up to grant or deny access to individual attribute types. In prior releases, access could only be granted to attribute access classes (groups of attributes).

What this change affects

This change affects the format of values of the **aclEntry** attribute. The changed format is a super set of the format that was supported in prior releases. Access control lists created using prior releases continue to work as before.

Dependencies

None.

Coexistence considerations

In order to use this support, all LDAP servers that are using the same set of DB2 tables (TDBM backend database instance) must be capable of interpreting the additional format for values of the **aclEntry** attribute. When running in a sysplex or multi-server mode, all LDAP servers must be running at the z/OS V1R4 or later level before this support can be enabled. This support is enabled by modifying the DB_VERSION value for the TDBM backend databse instance. New TDBM backend database instances, created using the z/OS V1R4 LDAP or later server, will initialize the DB_VERSION value to allow the enhanced controls to be used in the newly defined TDBM backend databse instance. In order to use such a TDBM access backend database instance in a sysplex or multi-server mode with z/OS LDAP servers from previous releases, the DB_VERSION value must be set to 1.0.

This support also requires the coexistence PTF for z/OS V1R3 (OW54426).

Migration tasks

Refer to “Migrating TDBM databases” on page 119 for detailed information on modifying the DB_VERSION value for the TDBM backend database instance.

For more information

See Chapter 22, “Using access control,” on page 269 for more information on using the new grant/deny and attribute-level access control capabilities.

SDBM enhancements

Description

SDBM has been enhanced to support:

1. Setting and displaying the values of the new EIM segment in the RACF user profile.
2. Specifying SHARED and AUTOUID or AUTOGID when setting the UID or GID value in the OMVS segment in the RACF user or group profile.
3. Searching for all the RACF users or groups with a specified OMVS UID or GID value.
4. Using an internal schema instead of defining the attributes and object classes in external schems files.

SDBM has also changed the reason code that it returns when a client attempts to bind using a user who doesn’t exist, a user who is not fully defined, or an incorrect password. All these cases now return reason code R000104. To cover all these cases, the text for reason code R000104 has been changed to:

```
R000104 The password is not correct or the user id is not completely defined
 (missing password or uid).
```

What this change affects

The new RACF control of shared UID/GID values can now be used when adding or modifying a user or group using SDBM. The reason code returned when binding to SDBM has changed as described above.

Dependencies
Enhancements 2 and 3 require that RACF be at AIM Stage 2 and require additional RACF setup. See the *z/OS Security Server RACF Security Administrator's Guide* for more information.

Coexistence considerations
In previous releases, binding to SDBM with a user who doesn't exist or who is not fully defined returned:

```
R000103 The user id is not completely defined - missing password or uid.
```

It now returns R000104 as described above.

Migration tasks
The use of an internal schema by SDBM instead of external schema files has several effects:

1. SDBM no longer uses external schema files. Any **include** options for schema files should be removed from the configuration file.
2. An SDBM suffix should not contain an alias for an attribute. For example, an SDBM suffix cannot use the **surName** attribute (it can use the **sn** attribute instead). Also, an SDBM suffix can contain a case-sensitive attribute in the suffix, but SDBM ignores case when processing the suffix. Applications that check the reason code returned for when binding to SDBM may need to be modified to accommodate the reason code change described above.

For more information
See Chapter 16, "Accessing RACF information," on page 213 for more information on SDBM support.

LDAP IBM-entryuuid support

Description
The z/OS LDAP server provides support for **IBM-entryuuid**.

What this change affects
Adds an attribute which contains a unique identifier to every LDAP entry.

Dependencies
None.

Coexistence considerations
Only works with z/OS V1R3 with the coexistence APAR OW54426 and above z/OS LDAP servers.

Migration tasks
None.

For more information
See Chapter 11, "Running and using the LDAP backend utilities," on page 135 and the **serverEtherAddr** option on page 73.

Activity logging

Description
The z/OS LDAP server provides a means of collecting server activity information in a log file. The log file may be an HFS file or an MVS™ dataset.

What this change affects
A new option has been added to the configuration file to allow specification of the log file. New modify commands are supported by the z/OS LDAP server to control generation of the log file.

Dependencies
None.

Coexistence considerations
None.

Migration tasks
None.

For more information
For more information, see “Activity Logging” on page 113

Abandon support

Description
The Abandon operation was supported in previous releases of the LDAP server. Hoever, all operations on a single connection were processed synchronously. Therefore, the abandon operation had no effect on the processing or previous operations by the LDAP server. In z/OS V1R4, and later releases, the LDAP server reads additional operations as they arrive as long as the connection is not a secure connection and the previous operation is not bind, unbind, or extended operations.

What this change affects
None.

Dependencies
None.

Coexistence considerations
None.

Migration tasks
None.

For more information
None.

RDBM removal

Description
Support for the RDBM backing store is removed. Customers using this DB2-based backing store must migrate from use of RDBM to use of TDBM, which is also DB2 based.

What this change affects
TDBM allows storage of larger amounts of directory data and provides improved performance over RDBM.

Dependencies
None.

Coexistence considerations
Since RDBM is not available in z/OS V1R4 and later LDAP, customers wishing to add z/OS V1R4 or later release to a sysplex where they are already using the LDAP server and sharing an RDBM database across the sysplex must migrate their shared database to TDBM across the sysplex before adding z/OS V1R4 or a later release into the sysplex. TDBM is available in all z/OS releases. It is not possible to share data across a sysplex between an RDBM backing store and a TDBM backing store.

Migration tasks
To migrate from RDBM to TDBM, customers must unload their RDBM backing store to LDIF and reload that LDIF into a newly-defined TDBM backing store.

For more information
See “Coexistence and migration with previous releases (RDBM)” on page 120 for more information.

Monitor support

Description
The LDAP server provides the capability to retrieve certain statistics about its operations through an LDAP server using a base of `cn=monitor`.

What this change affects
Provides additional monitoring capability.

Dependencies
None.

Coexistence considerations
None.

Migration tasks
None.

For more information
See “Monitor Support” on page 336 for more information.

TDBM schema

Description
Additional attribute types and object classes are added to the TDBM schema by applying the updates from the **schema-update.ldif** file shipped with this release. Enhanced schema migration instructions are now available.

What this change affects
The TDBM schema is enhanced by the addition of the attribute types and object classes. Future updates to the schema will be simplified.

Dependencies
None.

Coexistence considerations
None.

Migration tasks
Current users if the TDBM backend who have not yet applied APAR OW53447 need to perform the schema upgrade tasks defined in Chapter 14, “LDAP directory schema,” on page 165.

For more information
See “Schema migration” on page 118 for additional information about migrating the schema.

LDAP server message enhancements

Description
Changed the severity level to be consistent with other z/OS product message severities. Modified the LDAP server message numbers to correctly reflect the severity level documented in the message description.

What this change affects
The message numbers have changed.

Dependencies
None.

Coexistence considerations
None.

Migration tasks
Use of message numbers in automation of LDAP events should be reviewed and modified to be consistent with the new message number values..

For more information
See Chapter 31, “LDAP server messages,” on page 377 for more information.

Chapter 11. Running and using the LDAP backend utilities

Utility programs are provided to assist in initializing and backing up the data managed by the LDAP server.

Operation	TDBM utility
Load data into backend database	**ldif2tdbm**
Unload data from backend database to an LDIF file	**tdbm2ldif**
Add IBM entry UUIDs to entries in a backend database	**ldapadduuids**
Encrypt passwords in a backend database	**db2pwden**

These programs can be run in the z/OS shell, as jobs using JCL and procedures, or from TSO.

Format and usage information for the utilities are in:
- "ldif2tdbm utility" on page 138
- "tdbm2ldif utility" on page 148
- "ldapadduuids utility" on page 151
- "db2pwden utility" on page 154

See "Creating the DB2 database and table spaces for TDBM or GDBM" on page 42 for information on the permissions necessary to run these utilities.

Running the LDAP TDBM backend utilities in the z/OS shell

In order to run the **ldif2tdbm**, **tdbm2ldif**, **ldapadduuids**, or **db2pwden** utilities in the shell, some environment variables need to be set properly. Sample shell scripts are shipped with the LDAP server for **ldif2tdbm**, **ldapadduuids**, and **tdbm2ldif**. The sample shell scripts are located in **/usr/lpp/ldap/sbin**. These shell scripts need to be modified to fit your environment. The **PATH** variable should be set in the shell scripts. Ensure that **/usr/sbin** is added to the **PATH** environment variable. You should set **STEPLIB** to SYS1.SIEALNKE, if the PDS has not been placed in LINKLIST. You should also specify *DB2HLQ*.SDSNLOAD in the **STEPLIB**, if this PDS has not been placed in the LINKLIST or LPA.

If you are using the **ldif2tdbm** utility to add an entry containing a password that will be encrypted, the **LIBPATH** variable may need to be set. See "Installing OCSF and ICSF for password encryption" on page 16 for more information.

When started, **db2pwden**, **ldif2tdbm**, and **tdbm2ldif** read an environment variable file. The default file is **/etc/ldap/slapd.envvars**. This default can be changed by setting the environment variable **LDAP_SLAPD_ENVVARS_FILE** to the full path name of the desired environment variable file. Some of the environment variables that can be set are **NLSPATH** and **LANG**.

Running the LDAP TDBM backend utilities from JCL

Sample JCL for running **ldapadduuids**, **db2pwden**, **ldif2tdbm**, and **tdbm2ldif** from batch is provided with the LDAP server. The JCL includes an inline procedure, which will need to be modified by each installation to ensure that the appropriate load modules can be found. It may also be necessary to modify the **JOB** card for installation-specific requirements. These jobs can be run by editing the JCL member of the *GLDHLQ*.SGLDSAMP dataset and entering the **submit** command.

Notes:

1. If your runtime libraries for DB2 are not in LINKLIST or LPA on the system, make sure you specify the DB2 high-level qualifier for your DB2 installation in a STEPLIB DD card in the LDF2TDBM or TDBM2LDF batch job. These two utilities require the following DB2 dataset:

DB2HLQ.SDSNLOAD

Running the LDAP TDBM backend utilities in TSO

The utilities can be run from TSO. Following are the steps to do this:

1. Make the PDS (*GLDHLQ*.SGLDEXEC) containing the CLISTs needed to run the utilities available in SYSEXEC:

   ```
   alloc f(SYSEXEC) da('GLDHLQ.SGLDEXEC')
   ```

2. You can change the default environment variable file for the utilities by creating a dataset to hold the environment variables and then using the TSO **alloc** command as shown:

   ```
   alloc f(ENVVAR) da('datasetname')
   ```

If you want to specify a configuration file that is a data set you can specify the **-f** option on the command. For example:

```
tdbm2ldf -f "//'datasetname'" -o /tmp/ldif.1
```

Alternately, to specify the configuration file by associating it with a DD name, enter in TSO:

```
alloc da('datasetname') fi(config) shr
```

and then invoke the utility without specifying the **-f** option.

Once this setup is complete, running these utilities follows the same syntax as would be used if running in the z/OS shell, except you must use **ldf2tdbm** instead of **ldif2tdbm**, **tdbm2ldf** instead of **tdbm2ldif**, and **ldapuuid** instead of **ldapadduuids**. There is no difference with **db2pwden**. See “Running the LDAP TDBM backend utilities in the z/OS shell” on page 135.

SSL/TLS information for LDAP utilities

The contents of a client’s key database file is managed with the **gskkyman** utility. See *z/OS System Secure Sockets Layer Programming Guide* for information about the **gskkyman** utility. The **gskkyman** utility is used to define the set of trusted certification authorities (CAs) that are to be trusted by the client. By obtaining certificates from trusted CAs, storing them in the key database file, and marking them as trusted, you can establish a trust relationship with LDAP servers that use certificates issued by one of the CAs that are marked as trusted.

If the LDAP servers accessed by the client use server authentication, it is sufficient to define one or more trusted root certificates in the key database file. With server authentication, the client can be assured that the target LDAP server has been issued a certificate by one of the trusted CAs. In addition, all LDAP transactions that flow over the SSL/TLS connection with the server are encrypted, including the LDAP credentials that are supplied on the **ldap_bind** API.

For example, if the LDAP server is using a high-assurance VeriSign certificate, you should obtain a CA certificate from VeriSign, receive it into your key database file, and mark it as trusted. If the LDAP server is using a self-signed **gskkyman** server certificate, the administrator of the LDAP server can supply you with a copy of the server’s certificate request file. Receive the certificate request file into your key database file and mark it as trusted.

Using the LDAP utilities without the **-Z** parameter and calling the secure port on an LDAP server (in other words, a non-secure call to a secure port) is not supported. Also, a secure call to a non-secure port is not supported.

SSL/TLS encrypts the keyring file. Either the password must be specified as part of the **-P** parameter or file specification of a stash file that was created using the **gskkyman** utility must be specified in the form *file://* followed immediately (no blanks in between) by the file specification of the stash file.

Using RACF key rings

Alternately, LDAP supports the use of a RACF key ring. See the certificate/key management section in *z/OS Crytographic Services System Secure Sockets Layer Programming* for instructions on how to migrate a key database to RACF and how to use the **RACDCERT** command to protect the certificate and key ring.

The user ID under which the LDAP client runs must be authorized by RACF to use RACF key rings. To authorize the LDAP client, you can use the RACF commands in the following example (where *userid* is the user ID running the LDAP client utility):

```
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(userid) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(userid) ACCESS(CONTROL)
```

Remember to refresh RACF after doing the authorizations.

```
SETROPTS RACLIST(FACILITY) REFRESH
```

Once the RACF key ring is set up and authorized, specify the RACF key ring name for the **-K** *keyfile* option and do not specify the **-P** *keyfilepw* option.

ldif2tdbm utility

Purpose

This program is used to load entries specified in LDAP Data Interchange Format (LDIF) into a TDBM directory stored in a relational database. **ldif2tdbm** cannot be used to load entries into a GDBM database. The TDBM database must already exist. The **ldif2tdbm** program is intended for loading a large number of entries. The utility creates load records from the entries in the LDIF input files, then runs the DB2 Load Utility to load the records into the TDBM database.

The **ldapadd** command can also be used to add entries to a TDBM database. See "When to use ldif2tdbm" on page 142 for more information on when to use **ldif2tdbm** or **ldapadd**. See *z/OS Integrated Security Services LDAP Client Programming* for more information on **ldapadd**.

The **ldif2tdbm** program may be used to add entries to an empty directory database, or to a database that already contains entries. The **ldif2tdbm** utility may also be used to modify the schema entry before loading new entries into the database.

See "Preparing to run ldif2tdbm" on page 140 before using **ldif2tdbm**.

Format

```
ldif2tdbm {[-c [-n noe|nop|noep]] [-p] [-l]}
-o outHlq {[-i ldifFile[,ldifFile]...] [-e ldifListFile]}
[-s schemaLdifFile[,schemaLdifFile]...] [-v schemaListFile] [-f confFile] [-a yes|no] [-t logFile]
[-b creatorDN] [-q summaryFrequency] [-d debugLevel]
```

Parameters

-c Check that the entries in the LDIF input file or files are complete and acceptable according to the current internal schema. If the **-s** option is specified, the current internal schema is updated using the entries in the schema LDIF files before the LDIF entries are checked.

-n noe | nop | noep
Control the amount of checking that is done for each LDIF entry. In some cases, the checking that is skipped is performed anyway during the prepare step. See the specific information after each value.

- **noe** - Do **not** perform entry existence checks:
  - Do not check that the entry does not already exist, either as a previous LDIF entry or in the LDAP database.

  Note that a previous duplicate LDIF entry will be detected and rejected during the prepare step (when **-p** is specified). However, if **noe** is specified, an existing entry in the database will not be detected. Thus the load step (when **-l** is specified) can result in a duplicate entry in the LDAP database.
- **nop** - Do **not** perform several parent checks:
  - Do not check that the entry's parent exists, either as a previous LDIF entry or in the LDAP database.
  - Do not check that the LDIF entry is not under a referral entry or under an alias entry.

  Note that the checks skipped by **nop** are always performed during the prepare step (when **-p** is specified).
- **noep** - Do **not** perform these checks:
  - Do not check that the LDIF entry already exists.
  - Do not check that the entry's parent exists.
  - Do not check that the entry is not under a referral entry or under an alias entry.

-p Prepare DB2 table load files and JCL from the entries in the LDIF input file or files. The load and JCL files are generated as datasets, whose high-level qualifier is specified with the **-o** option. The prepare

step deletes the existing contents of the datasets before writing new contents to the datasets. If the **-s** option is specified, the current internal schema is updated using the entries in the schema LDIF files before the LDIF entries are prepared.

-l Invoke the DB2 Load Utility to load the LDAP database. Also, update the schema entry in the database if **-s** is specified.

-o *outHlq*
Specify the high level qualifier of the datasets that will contain the DB2 load and JCL output, the status information, and the system information. These datasets must be allocated before invoking **ldif2tdbm**. See “Preparing to run ldif2tdbm” on page 140 for more information.

-i *ldifFile*
Specify the name of an LDIF file to use as input. If a list of file names is specified, each file in the list is processed in turn. If **ldif2tdbm** is invoked separately to run each **ldif2tdbm** step (check, prepare, and load), make sure to specify the same set of LDIF files each time you run **ldif2tdbm**. The **ldif2tdbm** program issues a warning prompt (unless **-a** is specified) if the list of LDIF file names is different.

See “Using LDIF format to represent LDAP entries” on page 328 for more information on the general format of LDIF entries.

-e *ldifListFile*
Specify the name of a file which contains a list of LDIF input files to be used as input. This is equivalent to using the **-i** option to specify a list of LDIF files, but is more convenient for a large list. Each record in the list file must contain the name of one LDIF input file. Blank lines and lines beginning with a # (comment lines) are ignored. See the **-i** explanation above for more information about using a list of LDIF files.

-s *schemaLdifFile*
Specify the name of an LDIF file containing only schema entries. The entries must be in **ldapmodify** LDIF-mode format (see Note 8 on page 146). If a list of file names is specified, each file in the list is processed in turn. During the check or prepare steps (when **-c** or **-p** is specified), the current internal schema is updated using these schema files, but the schema is not changed in the database. This updated internal schema is used to check the entries in the LDIF input files or to prepare the load files. During the load step (when **-l** is specified), these schema files are used to update the current schema in the database. If **ldif2tdbm** is invoked separately to run each **ldif2tdbm** step (check, prepare, and load), make sure to specify the same set of LDIF schema files each time you run **ldif2tdbm**. The **ldif2tdbm** program issues a warning prompt (unless **-a** is specified) if the list of LDIF file names is different.

-v *schemaListFile*
Specify the name of a file which contains a list of LDIF input files to be used to modify the schema entry. This is equivalent to using the **-s** option to specify a list of LDIF files, but is more convenient for a large list. Each record in the list file must contain the name of one LDIF input file. Blank lines and lines beginning with a # (comment lines) are ignored. See the **-s** explanation above for more information about using a list of LDIF files.

-f *confFile*
Specify the name of the LDAP configuration file to use. This configuration file only needs to contain information for the TDBM backend into which the entries are to be loaded. The default is **/etc/ldap/slapd.conf**.

-a yes|no
Eliminate the prompt that **ldif2tdbm** issues when it detects an unexpected status condition by providing an answer to the prompt. If an unexpected status condition is encountered, **ldif2tdbm** continues if **-a yes** is specified and exits if **-a no** is specified. If **-a** is not specified, **ldif2tdbm** issues a prompt and waits for a response. See Note 11 on page 146 for more information on status checking.

-t *logFile*
Specify the name of the file to which messages are written. If **-t** is not specified, messages are sent to **stdout** or **stderr**. The existing contents of the file are deleted.

-b *creatorDN*
The DN to be associated as creator with each loaded entry that does not already include a **creatorsname** attribute. If **-b** is not specified, the value of the **adminDN** option in the LDAP server configuration file is used. If neither option is specified, **ldif2tdbm** fails. The same processing is also applied for the modifier for each loaded entry that does not already include a **modifiersname** attribute.

-q *summaryFrequency*
Specify the number of LDIF entries the prepare step should process between issuing summary messages. The default value is 1000, resulting in issuing a summary message after every 1000 entries are prepared. If you have many LDIF entries, increase this value to reduce the frequency of summary messages. Specify a negative value or 0 to suppress issuing any intermediate summary messages. A final summary message is always issued.

-d *debugLevel*
Specify the level of debug messages to be created. The debug level is specified in the same fashion as the debug level for the LDAP server, as described on page 102. Table 18 on page 102 lists the specific debug levels. The default is no debug messages.

All other command line inputs will result in a syntax error message, after which the proper syntax will be displayed. Also, specifying the same option multiple times with different values results in a syntax error message.

Examples

Following is an example using the **ldif2tdbm** utility:

```
ldif2tdbm -cpl -i /data2/ldif.data -s /data2/schema_changes.data -o admin3.prv -d ERROR
```

This **ldif2tdbm** utility invocation checks, prepares, and loads the LDIF data from `/data2/ldif.data`, updates the schema using the LDIF data from `/data2/schema_changes.data`, uses the output datasets `ADMIN3.PRV.BULKLOAD.INPUT.xxx` and `ADMIN3.PRV.BULKLOAD.JCL`, and specifies a debug level of ERROR for **LDAP_DEBUG_ERROR**. By default, the configuration file used is **/etc/ldap/slapd.conf**, complete checking is done on the entries, the user is prompted whether to continue if warning conditions are detected, messages are sent to **stdout** or **stderr**, the value of the **adminDN** option in the configuration file is used as the creator of each loaded entry, and a progress message is issued after every 1000 entries processed.

Preparing to run ldif2tdbm

Before invoking **ldif2tdbm**, you must

- Allocate the output datasets used by **ldif2tdbm**.
- Create the SYSTEM file used by the prepare step of **ldif2tdbm**.
- Set the environment variables used by **ldif2tdbm**.

Allocating datasets required by ldif2tdbm

The various **ldif2tdbm** processing steps use 5 load record datasets and a JCL dataset. These datasets must be allocated before invoking **ldif2tdbm** and the high-level qualifiers in the dataset names must be specified as the value of the **-o** *outHlq* option on the command. The **ldif2tdbm** utility writes the contents of these datasets, except for the SYSTEM member of the JCL dataset which must be created before **ldif2tdbm** is run.

- Load Record Datasets

  The prepare step of **ldif2tdbm** writes the load data created from each LDIF entry into the load record datasets. Each dataset contains the records for one database table and is used as input to the DB2 Load Utility when loading that table.

  – Dataset names:
  outHlq.BULKLOAD.INPUT.DESC (DIR_DESC load dataset)

outHlq.BULKLOAD.INPUT.ENTRY (DIR_ENTRY load dataset)
outHlq.BULKLOAD.INPUT.LATTR (DIR_LONGATTR load dataset)
outHlq.BULKLOAD.INPUT.LENTRY (DIR_LONGENTRY load dataset)
outHlq.BULKLOAD.INPUT.SEARCH (DIR_SEARCH load dataset)

- Dataset format:
  Sequential (non-PDS)
  Record format = VB
- Dataset record length and block size:

  The record length depends on the page size of the corresponding table space. The actual record length may be reduced slightly by **ldif2tdbm** at run-time.

  For a 32K page size in DB2, use `LRECL=32756, BLKSIZE=32760`.

  This record length and block size will also work for smaller page sizes. However, for smaller page sizes, a smaller LRECL and BLKSIZE can be used to reduce the required disk space. For example, on 3390 DASD, `LRECL=27994, BLKSIZE=27998` will allow 2 blocks per track and, in general, more bytes per track to be written. For the smaller page sizes, the LRECL and BLKSIZE must be at least pagesize + 6 and pagesize + 10, respectively.
- Dataset size:

  You may use the space estimation tool to determine the approximate size of the load data sets. This tool can be downloaded by selecting “Download” on:

  `http://www.ibm.com/software/network/directory`

  The tool is written in **awk** script and can be run under z/OS UNIX System Services. The script can be modified for your specific needs based on values specific to your configuration.

  Alternatively, a rough estimate for the size (in bytes) of each dataset is as follows:

  outHLQ.BULKLOAD.INPUT.DESC:
  34 * (average depth) * (number of entries in the LDIF files)

  where average depth = (average number of levels in a DN) - (average number of levels in each DN’s suffix) + 1

  outHLQ.BULKLOAD.INPUT.ENTRY, *outHLQ*.BULKLOAD.INPUT.LATTR, and *outHLQ*.BULKLOAD.INPUT.LENTRY:
  The combined space required for these files is roughly

  (number of bytes in the LDIF input files) * 3.0

  If most directory entries are shorter than the DIR_ENTRY page size, and most attributes are shorter than the **attrOverflowSize** in the LDAP server configuration file, then most of the data will be written in the *outHLQ*.BULKLOAD.INPUT.ENTRY dataset. Otherwise, you may wish to allocate each of these three datasets as this maximum size to ensure that you do not run out of space.

  outHLQ.BULKLOAD.INPUT.SEARCH:

  (number of bytes in the LDIF input files) * 2.5

- JCL dataset

  The JCL dataset is a PDS whose members contain system information, status information, and the JCL to run DB2 Load Utility for each database table that needs to be loaded. All steps of **ldif2tdbm** use the status information. The prepare step writes the contents of the JCL members of the JCL dataset, using the information in the system member. The system member of the JCL dataset must be created by the user before **ldif2tdbm** is invoked to run the prepare step.
  - Dataset name:
    outHlq.BULKLOAD.JCL
  - Members:
    outHlq.BULKLOAD.JCL(JDESC) (DIR_DESC load JCL)
    outHlq.BULKLOAD.JCL(JENTR) (DIR_ENTRY load JCL)
    outHlq.BULKLOAD.JCL(JLATT) (DIR_LONGATTR load JCL)

outHlq.BULKLOAD.JCL(JLENT) (DIR_LONGENTRY load JCL)
outHlq.BULKLOAD.JCL(JSRCH) (DIR_SEARCH load JCL)
outHlq.BULKLOAD.JCL(STATUS) (Status information)
outHlq.BULKLOAD.JCL(SYSTEM) (System information - see below)

- Dataset format:
  Partitioned (PDS)
  Record format = FB
  Record length = 80
- Dataset size:
  200K bytes

Creating the SYSTEM file

The contents of the SYSTEM file, *outHlq*.BULKLOAD.JCL(SYSTEM), must be created before invoking **ldif2tdbm** to run the prepare step, which uses the information in the SYSTEM file to create the JCL to invoke the DB2 Load Utility to load each of the database tables.

- Format of SYSTEM records

```
SSID yyy              Subsystem ID for DB2
HLQ db2hlq            High level qualifier for DB2 datasets
JOBCARD //jobname...  Job card record for JCL
#Comment record       Ignored if first non-blank character is #
```

- The SYSTEM member must contain one SSID record, one HLQ record, and one or more JOBCARD records. The first JOBCARD record must begin with //*jobname* where *jobname* is at most 8 characters. The maximum length for each record value is 55 for the SSID value, 36 for the HLQ value, and 71 for each JOBCARD value. Make sure there are no sequence numbers at the end of each line.
- To differentiate the load jobs in the 5 JCL members of the JCL PDS, **ldif2tdbm** replaces the last character of the job name with the digits 1 through 5, as follows:
  Jobname ends in 1 - JCL for loading DIR_DESC table, in *outHlq*.BULKLOAD.JCL(JDESC)
  Jobname ends in 2 - JCL for loading DIR_ENTRY table, in *outHlq*.BULKLOAD.JCL(JENTRY)
  Jobname ends in 3 - JCL for loading DIR_LONGATTR table, in *outHlq*.BULKLOAD.JCL(JLATT)
  Jobname ends in 4 - JCL for loading DIR_LONGENTRY table, in *outHlq*.BULKLOAD.JCL(JLENT)
  Jobname ends in 5 - JCL for loading DIR_SEARCH table, in *outHlq*.BULKLOAD.JCL(JSRCH)

Setting environment variables used by ldif2tdbm

The **ldif2tdbm** utility uses two environment variables. These environment variables are optional, but if set properly, they may increase performance.

- **LDIF2TDBM_CACHE_SIZE**

  The **LDIF2TDBM_CACHE_SIZE** environment variable sets the number of internal structures that **ldif2tdbm** should allocate. This number should be set to two times the number of entries that will be loaded.

- **LDIF2TDBM_LOG_DB**

  The **LDIF2TDBM_LOG_DB** environment variable determines whether **ldif2tdbm** will force the DB2 Load Utility to do logging as it loads the database. See Note 5 on page 143 for more information.

When to use ldif2tdbm

Both the **ldif2tdbm** utility and the **ldapadd** command can be used to load entries into a TDBM database. There are advantages and disadvantages to each of these. Following is a list of considerations that can assist in determining which method to use when loading entries.

Table 21. Considerations for using ldif2tdbm or ldapadd

Considerations	ldif2tdbm	ldapadd
Speed of load	Faster	Slower, especially if the database is already large.

Table 21. Considerations for using ldif2tdbm or ldapadd (continued)

Considerations	ldif2tdbm	ldapadd
Operational attributes	Accepts input containing **creatorsname**, **modifiersname**, **createtimestamp**, **modifytimestamp**, and **ibm-entryuuid** operational attributes.	Input cannot contain these operational attributes.
Complexity	High, will take time to learn. Normal usage requires multiple invocations, with review of JCL and preparation for recovery before load.	Low
Set up	User must allocate datasets and create the SYSTEM information file before running **ldif2tdbm** for the first time.	No set up.
LDAP server down time	LDAP server must be down during load step. Server can be up during check and prep steps.	Server must be operational during adds.
DB2 logging	Logging is optional. Additional DB2 work is needed to make database fully usable after the load if logging is not done.	Requires logging.
Recovery	Recovery from load failure is complex, involving knowledge of DB2 Utilities.	Simple recovery.
Invocation	Must be run from z/OS system containing the database, or any image in a Parallel Sysplex running a DB2 subsystem which is a member of the DB2 data sharing group containing the database, using a user ID with DB2 privileges.	Can be run from any LDAP client with appropriate LDAP access.
Re-usability	Prepared entries are saved, so they can be re-used to load another system or reload this system.	No saved output.
Replication	New entries are not replicated.	New entries can be replicated.

Summary: The **ldif2tdbm** utility usage is complex, but it is fast. The **ldapadd** utility usage is simpler, but it is slower.

Recommendation: For one-time additions of 100K or more entries, or for frequent additions of 10K or more entries, use **ldif2tdbm**. For infrequent additions of less than 10K entries, use **ldapadd**. For additions of between 10K and 100K entries, use either **ldif2tdbm** or **ldapadd**.

ldif2tdbm performance considerations

The **ldif2tdbm** utility performance considerations are:

1. **ldif2tdbm** uses a lot of storage when adding a large number of entries. Make sure you have sufficient memory available.
2. If the parent of an entry being added is in the TDBM database, then **ldif2tdbm** must also check the database to ensure that the entry itself does not already exist. Thus, to minimize the database checks, include the parents of the entries being added in the **ldif2tdbm** input whenever possible. For example, do not use **ldapadd** to add a suffix and then use **ldif2tdbm** to add all the entries under the suffix. Instead, include the suffix in the **ldif2tdbm** input.
3. Do not specify the debug command line option, **-d**. Usage of debug can impact performance even when there is little debug output. Only specify **-d** when an error has occurred that you cannot fix.
4. If you know that your LDIF input is acceptable, consider using the **-n** option to limit the amount of checking done during the check (**-c**) step. You can also combine the check (**-c**) and prepare (**-p**) steps rather than doing them separately. In particular, using the **-cp -n noe** options will reduce the checks that access the database. This is especially useful if the parents of the entries being added are not in the **ldif2tdbm** input (see 2 above).
5. By default, for better load performance **ldif2tdbm** sets the **LOG NO** option in the DB2 Load Utility JCL created during the prepare (**-p**) step. When the load (**-l**) step invokes the DB2 Load Utility, the Load

Utility will set the copy pending restriction against the table space. The database can be read but cannot be updated until the restriction is removed, for example, by running the DB2 **REORG** or **COPY** utility. To avoid entering the copy pending state (for example, when the database already contains a large number of entries), change **LOG NO** to **LOG YES** in the JCL. This will be done by **ldif2tdbm** if the **LDIF2TDBM_LOG_DB** environment variable is set to 1 before running the prepare step. It can also be done manually in each of the *outHLQ*.BULKLOAD.JCL members after the prepare step if the load step is run separately from the prepare step.

ldif2tdbm normal usage

The normal usage of **ldif2tdbm** is:

1. Perform the necessary setup for running **ldif2tdbm**, as described in the "Preparing to run ldif2tdbm" on page 140. This consists of:
   - allocating datasets required by **ldif2tdbm**
   - creating the SYSTEM member in the *outHLQ*.BULKLOAD.JCL dataset
   - exporting the **LDIF2TDBM_CACHE_SIZE** and **LDIF2TDBM_LOG_DB** environment variables, if desired.

   Note that the same datasets can be used for loading different entries, but the contents of the datasets will be overwritten each time.
2. Repeatedly invoke **ldif2tdbm** with only the check (**-c**) option until all problems in the LDIF input files have been resolved. The check step can be skipped if you are sure that the LDIF input files contain only valid entries; however, although the prepare step does some checking of each entry, it might not detect all problems and might allow an entry that is not valid to be added to the database.
3. Run **ldif2tdbm** with the prepare (**-p**) option to prepare the load data.
4. Bring the LDAP server down.
5. Even if loading an empty database, make a full image copy of the table spaces for the DIR_DESC, DIR_SEARCH, DIR_ENTRY, DIR_LENTRY, and the DIR_LATTR tables. A full image copy is required to help recover from any potential DB2 Load Utility failures. It is done after the **ldif2tdbm** invocation with the prepare (**-p**) option specified because the prepare (**-p**) option may update some of the tables **ldif2tdbm** is attempting to load. Thus, these updates must be captured for a successful recovery from a DB2 Load Utility failure. If the prepare (**-p**) and load (**-l**) options are specified together, the full image copy should be done before the **ldif2tdbm** invocation.
6. Review the JCL created by the prepare step in the JCL dataset, *outHLQ*.BULKLOAD.JCL. Ensure that the DB2 Load Utility work datasets are large enough and that the DB2 Load Utility options, especially the **LOG** value, are acceptable. See note 5 in the ldif2tdbm performance considerations section for more information on the **LOG** value.
7. Run **ldif2tdbm** once more, with the load (**-l**) option to load the data into the database. This will submit 5 batch jobs to load the database.
8. Review the output from the load jobs when they terminate. If the loaded table spaces are in the copy pending state, refer to *DB2 Utility Guide and Reference* for instructions on removing that restriction on table spaces. This usually involves running a DB2 utility to create an image copy of the database or reorganize the database (or both).
9. Run the DB2 **runstats** utility to reset the statistics used by DB2 to access the database.
10. Run the DB2 **copy** utility to make an backup image copy of the database.

ldif2tdbm recovery

The **ldif2tdbm** program can determine whether the submission of the DB2 Load Utility jobs was successful, but it cannot determine if the DB2 Load Utility jobs themselves succeed. In fact, **ldif2tdbm** normally terminates before the jobs are finished. Thus, the final **ldif2tdbm** success message will appear even if one or more of the jobs eventually fails.

If a DB2 Load Utility job fails, terminate all of the DB2 Load Utility invocations that failed, using `-TERM UTIL(BULKx)`

where *x* is the last number in the job name of the failing job. Then, there are two alternatives for recovery.

Recovery process 1

The first recovery alternative is not selective in nature and may result in unnecessary reloading of data. This alternative must be used for recovery when:

- The **ldif2tdbm** utility is run with the load (**-l**) option immediately after creating the TDBM database (that is, without first starting the server or running **ldif2tdbm** with the check (**-c**) or prepare (**-p**) options).
- Loading the DIR_ENTRY, DIR_LENTRY, or DIR_LATTR table fails and a full image copy of these tables does not exist.

Following is the first recovery alternative process:

1. If full image copies exist for all 5 table spaces, use the DB2 Recover Utility to recover all 5 table spaces from those full image copies. Otherwise, drop and recreate the DIR_ENTRY, DIR_LATTR, DIR_SEARCH, DIR_DESC, and DIR_LENTRY tables.
2. Refer to *DB2 Utility Guide and Reference* for information on correcting the problems logged in the failing jobs.
3. Run **ldif2tdbm** with only the load (**-l**) option again.

Recovery process 2

The second recovery alternative requires less processing, but requires a greater understanding of the problem. It cannot be used for recovery when:

- The **ldif2tdbm** utility is invoked with the load (**-l**) option immediately after creating the TDBM database.
- Loading the DIR_ENTRY, DIR_LENTRY, or DIR_LATTR table fails and a full image copy of these tables does not exist.

Following is the second recovery alternative process:

- If the load jobs that failed involve only the DIR_SEARCH or DIR_DESC tables, follow these steps:
  1. If a full image copy exists for the failing table spaces, use the DB2 Recover Utility to recover these table spaces from those full image copies. Otherwise, drop and recreate the failing tables.
  2. Refer to *DB2 Utility Guide and Reference* for information on correcting problems logged in the failing jobs.
  3. Manually resubmit the DB2 Load Utility jobs that failed.
  4. Make sure that the DIR_DESC table contains a row with the values (-2, -2).
- If the load jobs that failed involve the DIR_ENTRY, DIR_LENTRY, or DIR_LATTR tables and a full image copy exists (you must use the first recovery alternative if a full image copy does not exist), follow these steps:
  1. If a schema file was specified using the **-s** option, the DIR_ENTRY, DIR_LENTRY, and DIR_LATTR tables must **all** be recovered, along with other failing tables. If a schema file was not specified, only the failing tables need to be recovered.
  2. If a full image copy exists for the affected table spaces, use the DB2 Recover Utility to recover these table spaces from those full image copies. Otherwise, you must use the first alternative.
  3. Refer to *DB2 Utility Guide and Reference* for information on correcting problems logged in the failing jobs.
  4. Manually resubmit the DB2 Load Utility jobs for all the affected tables.
  5. Make sure that the DIR_DESC table contains a row with the values (-2, -2).
  6. If a schema file was specified using the **-s** option, follow these steps:
     a. If the table spaces are in the copy pending state, refer to *DB2 Utility Guide and Reference* for information on removing that restriction on the table spaces.
     b. Start the LDAP server.

c. Remove the `'version: 1'` record from the schema file, if it contains one. Then, perform an LDAP modify using the schema file as input.

Usage

1. All input files specified with the **-i**, **-e**, **-s**, and **-v** options can be Unix file system files (for example: HFS) or datasets. In order to use datasets with **ldif2tdbm**, you must use a VB record format. Further, separator lines between directory entries in the LDIF file must contain only space characters (X'40'), if anything.
2. The **ldif2tdbm** utility can be invoked to run the check (**-c**) and prepare (**-p**) steps while an LDAP server using the same TDBM database is active. The LDAP server **must be down** when the load (**-l**) step is run.
3. The **LDAP_DEBUG** environment variable can also be used to set the debug level for **ldif2tdbm**. See page 102 for more information on specifying the debug level.
4. The DB2 **runstats** utility should be run once data is loaded so that DB2 queries are optimized.
5. No replication is performed for entries added by **ldif2tdbm**. A new replica entry can be added using **ldif2tdbm**, but replication does not begin until the LDAP server is started.
6. Referral entries can be added by **ldif2tdbm**.
7. Only one TDBM database is loaded in an invocation of **ldif2tdbm**. All the LDIF entries in the LDIF input files and schema input file must contain DNs that belong to the same TDBM database. The parent of each LDIF entry must either be already in the database or must be a prior LDIF entry within these LDIF input files.
8. Restrictions for updating the schema directory entry using **ldif2tdbm:**
   - The schema can only be updated within an **ldif2tdbm** invocation by using the **-s** *schemaLdifFile* and **-v** *schemaListFile* options. The LDIF input files specified in the **-i** and **-e** options cannot contain a schema entry.
   - The LDIF schema files specified using the **-s** and **-v** options must contain only schema entries, in the LDIF mode format supported by **ldapmodify**, described in *z/OS Integrated Security Services LDAP Client Programming* with the following changes:
     – If the optional **changetype** line is specified, the value must be **modify**. No other **changetype** value is supported for the schema.
     – The change indicator line (`add:x`, `replace:x`, or `delete:x`) is not optional and there is no default change indicator. Each **change_clause** must begin with a change indicator line.
9. The **aclSource** and **ownerSource** attributes should not be specified in an LDIF entry and are ignored. These attributes are set only by the system.
10. The **ldif2tdbm** utility check processing is terminated after 100 syntax errors are detected. The **ldif2tdbm** prepare and load processing are terminated after the first error. These values cannot be modified.
11. Since normal **ldif2tdbm** usage can involve multiple invocations to check, prepare, and load the entries, **ldif2tdbm** maintains a status file to keep information about the last successful step processed. The status file is *outHlq*.BULKLOAD.JCL(STATUS), where *outHlq* is the value of the **-o** option on the command. Any invocation of **ldif2tdbm** with the same value for **-o** is considered a continuation of processing of an earlier invocation.
    - Before processing any entries, **ldif2tdbm** uses the information about the earlier invocation in the status file to determine that each processing step is performed in order, the input files used for this invocation are the same as the ones used for the earlier invocation, and this invocation is not going to delete output prepared by the earlier invocation.

      The **ldif2tdbm** utility issues a warning message for each condition that it detects, followed by a single prompt asking if processing should continue. The prompt can be suppressed by specifying the **-a yes|no** option on the command to provide an answer for the prompt.
    - At the end of processing, **ldif2tdbm** rewrites the status file with information about the last successful step processed (check, prepare, or load), except as follows:
      – If the check step succeeds but

- Only the check (**-c**) step is requested, or the check (**-c**) and load (**-l**) steps are requested and the load step fails
- The previous status is prepare (P) or load (L)

the status file is not rewritten, to avoid replacing a higher status with a lower one.

– If no step succeeds but
- The prepare (**-p**) step fails
- The previous status is prepare (P) or load (L)

the status in the status file is reset to none (N), since the prepare step has deleted the contents of the load and JCL datasets.

12. There is additional checking in the load (**-l**) step. The load step checks whether the schema currently in the database has been changed since the load files were prepared. If the schema has been changed, the load step is terminated to avoid loading entries prepared using an older schema that might not be valid for the newer schema.
13. During the check (**-c**) and prepare (**-p**) steps, the RDN of the new LDIF entry is checked to ensure that all the values specified in the RDN are also specified as attributes in the LDIF entry. Missing attributes are added to the entry before it is loaded into the directory. No messages are issued indicating this.
14. The **ldif2tdbm** program encrypts clear text **userPassword** attribute values for new entries loaded into the TDBM backend with the **pwEncryption** method specified in the configuration file. The **ldif2tdbm** program can load the LDIF format of an encrypted password unloaded by the **tdbm2ldif** program. The **ldif2tdbm** program cannot load the LDIF format unloaded by the **tdbm2ldif** program with the **-t** option. The **-t** option unloads encryption ″tag visible″ format passwords for use with non-z/OS LDAP servers. The **ldif2tdbm** program expects that textual data contained within the LDIF file is portable and of UTF-8 origin.
15. If you are maintaining multiple identical databases, you can use the prepare (**-p**) step of **ldif2tdbm** to prepare load data using one TDBM database and then invoke the load (**-l**) step of **ldif2tdbm** to load the data into a second TDBM database. When doing this, the contents of the second database must be identical to the first database at the time the load data was prepared, including the same schema entry in the database. After invoking the **ldif2tdbm** load (**-l**) step on the second database, you must also update the value of the NEXT_EID column in the DIR_MISC table in the second TDBM database. To do this, use the following SPUFI command to obtain the value of the NEXT_EID column in the first TDBM database after the **ldif2tdbm** prepare step is run:

    ```
    SELECT NEXT_EID FROM userid.DIR_MISC;
    ```

    Then use the following SPUFI command to update the NEXT_EID column in the second TDBM database to that value:

    ```
    UPDATE userid.DIR_MISC set NEXT_EID = value;
    ```
16. Each loaded entry has a **creatorsname**, **modifiersname**, **createtimestamp**, **modifytimestamp**, and **ibm-entryuuid** attribute. The values for these attributes can be specified in the LDIF input for the entry. If not:
    - The **creatorsname** and **modifiersname** attributes are assigned the value of the **-b** command line option if it is specified; otherwise, they are assigned the value of the **adminDN** option in the LDAP server configuration file.
    - The **createtimestamp** and **modifytimestamp** attributes are assigned a time set during utility initialization.
    - The **ibm-entryuuid** attribute is assigned a unique value generated by the utility.

tdbm2ldif utility

Purpose

This program is used to dump entries from a directory stored in a TDBM database into a file in LDAP Data Interchange Format (LDIF). **tdbm2ldif** cannot be used to unload a GDBM database.

Format

tdbm2ldif [-o *outputFile***] [-s** *subtreeDN* **| -n** *TdbmName***] [-f** *confFile***] [-d** *debugLevel***] [-t]**

Parameters

-o *outputFile*
Specify the output file to contain the directory entries in LDIF. All entries from the specified subtree are written in LDIF to the output file. If the file is not in the current directory, a fully-qualified name must be specified. If the **-o** option is not specified, then the output from the program is written to **stdout**.

-s *subtreeDN*
Identify the DN of the top entry of the subtree whose entries are to be converted to LDIF. This entry, plus all below it in the directory hierarchy, are converted and written to the output file. The **-s** option must be used to unload the schema entry. The **-s** option cannot be used when the **-n** option is specified.

-n *TdbmName*
Specify the name of the TDBM backend to unload. This is the name assigned to the backend on its database record in the configuration file. This can be used to indicate which TDBM backend to process when there are multiple TDBM backends in the configuration file. The **-n** option cannot be used when the **-s** option is specified.

-f *confFile*
Specify the name of the configuration file to use. This configuration file only needs to contain information for the TDBM backend which contains the entries to be converted to LDIF. Default is **/etc/ldap/slapd.conf**.

-d *debugLevel*
Specify the level of the debug messages to be created. The debug level is specified in the same fashion as the debug level for the LDAP server, as described on page 102. Table 18 on page 102 lists the specific debug levels. The default is no debug messages.

-t Specify that encrypted **userPassword** attribute values will be unloaded with their encryption tag in clear text, as follows:

```
userPassword: {tag}base64encoded_and_encryptedvalue
```

where *tag* is **none**, **crypt**, **MD5**, **SHA**, or **DES:***keylabel*.

Examples

```
userPassword: {none}321p90fa0fdvn;a

userPassword: {crypt}3sdfaf[a

userPassword: {SHA}24309gf[jgt

userPassword: {DES:kgup.data.key}3ajewomv..=
```

In this format, the tag is visible, and only the **userPassword** value itself is encrypted and base64 encoded.

Notes for using the -t parameter:

1. The format of data produced when **-t** is specified may be acceptable for other LDAP providers to load into their LDAP directory. And, if it is not directly loadable, this format is easily modified for loading by another provider into its LDAP directory. This format cannot be loaded back into an z/OS LDAP server.
2. The tag is enclosed by a left brace and a right brace. One colon is used between the **userPassword** keyword and the value, as opposed to two colons in the standard LDIF format of **userPassword** dumped by **tdbm2ldif**. This format cannot be read by the **ldif2tdbm** utility. It is intended for other LDAP providers and tools that may require the encryption tag visible.
3. Clear text passwords without a tag could still exist in the TDBM backend if the password was not modified or **pwEncryption** was not configured on the server. The values would be unloaded as standard binary attributes in base64 encoding. Following is an example:

   ```
   userPassword:: kfa6903axs
   ```
4. The values returned by the crypt() algorithm are not portable to other X/Open-conformant systems. This means that user password values encoded by the crypt() algorithm and unloaded as tagged output using **tdbm2ldif -t** are not portable when loaded by another platform's load utility.

If **-t** is not specified, the output for the **userPassword** attribute is in the format:

```
userPassword::base64encodedValue
```

where *base64encodedValue* is a base64 encoded value of the *binaryvalue*, and *binaryvalue*={*tag*}*encryptedPasswordValue*.

All other command line inputs will result in a syntax error message, after which the proper syntax will be displayed. Also, specifying the same option multiple times with different values will result in a syntax error message.

Examples

Following is an example using the **tdbm2ldif** utility:

```
tdbm2ldif -o /tdbmdata/ldif.data -n tdbm1 -f /ldap/conf/slapd.conf
```

This **tdbm2ldif** utility invocation dumps all the data, except the schema entry, from the TDBM backend named **tdbm1** in the configuration file to the file `/tdbmdata/ldif.data`, and uses the `/ldap/conf/slapd.conf` configuration file.

Usage

1. If the **tdbm2ldif** program is invoked with neither the **-s** nor the **-n** option and there is a single TDBM backend in the configuration file, all the directory entries in that TDBM databases are processed, except for the schema entry. If there are multiple TDBM backends in the configuration file, the program returns an error message.
2. The **tdbm2ldif** program only dumps owner and ACL information for entries that have a specific owner or ACL. Any entry data with an inherited owner or ACL will not have owner or ACL information dumped.
3. For the LDAP Version 3 protocol, there is a related set of Internet Drafts which discuss the introduction of a version mechanism for use in creating LDIF files. The **tdbm2ldif** utility always creates "tagged" LDIF files. The LDIF tag consists of a single line at the top of the LDIF file:

   ```
   version: 1
   ```

   All characters contained in the `version: 1` LDIF file are portable characters represented in the local codepage. Strings containing nonportable characters (for instance, textual values containing multi-byte UTF-8 characters) must be base64 encoded.
4. To unload the schema entry from a TDBM directory, specify

   ```
   -s cn=schema,suffixDN
   ```

   where *suffixDN* is the DN of a suffix in the TDBM directory. The schema entry is unloaded in LDIF modify format, which can be used in the **-s** option of the **ldif2tdbm** utility.

5. When editing the output file produced by **tdbm2ldif**, make sure to use an editor that does not delete blanks at the end of a line. If the output file contains a line that ends with blanks, using such an editor will result in deleting the blanks, thus changing the value of the attribute. This is especially serious when the line is continued, since the ending blanks will no longer be there to separate the last word in the line from the continuation on the next line. The maximum line length in a **tdbm2ldif** output file is 77; continued lines are always 77 characters long when the file is created by **tdbm2ldif**.

   The **oedit** editor is an example of an editor that deletes blanks and should **not** be used.
6. The **LDAP_DEBUG** environment variable can also be used to set the debug level for **tdbm2ldif**. See page 102 for more information on specifying the debug level.

ldapadduuids utility

Purpose

The **ldapadduuids** utility adds an **ibm-entryuuid** value to each entry in a LDAP TDBM directory that does not already have an **ibm-entryuuid** value. The **ibm-entryuuid** attribute holds a DCE standard Universally Unique Identifier (UUID). The UUID that is created is unique for all UUIDs on all computers. The **ibm-entryuuid** attribute holds a DCE standard Universally Unique Identifier (UUID). This utility allows you to migrate all or portions of an existing directory to contain UUIDs. Directory applications may make use of the UUID to identify entries.

Format

ldapadduuids [*options*] [*filter*]

Parameters

options

The following table shows the options you can use for the **ldapadduuids** utility:

-?	Print this text.
-S *method* or **-m** *method*	Specify the bind method to use. You can use either **-m** or **-S** to indicate the bind method. The default is **SIMPLE**. You can also specify **GSSAPI** to indicate a Kerberos Version 5 is requested, **EXTERNAL** to indicate that a certificate (SASL external) bind is requested, **CRAM-MD5** to indicate that a SASL Challenge Response Authentication Mechanism bind is requested, or **DIGEST-MD5** to indicate a SASL digest hash bind is requested. The **GSSAPI** method requires a protocol level of 3 and the user must have a valid Kerberos Ticket Granting Ticket in their credentials cache by using the Kerberos **kinit** command line utility. The **EXTERNAL** method requires a protocol level of 3. You must also specify **-Z**, **-K**, and **-P** to use certificate bind. If there is more than one certificate in the key database file, use **-N** to specify the certificate or the default certificate will be used. The **CRAM-MD5** method requires protocol level 3. The **-D** or **-U** option must be specified. The **DIGEST-MD5** method requires protocol level 3. The **-U** option must be specified. The **-D** option can optionally be used to specify the authorization DN.
-h *ldaphost*	Specify the host on which the LDAP server is running. The default is the local host. When the target host is a z/OS LDAP server operating in multi-server mode with dynamic workload management enabled (see), the *ldaphost* value should be in the form *group_name.sysplex_domain_name*, where *group_name* is the name of the **sysplexGroupName** identified in the server configuration file and *sysplex_domain_name* is the name or alias of the sysplex domain in which the target server operates.
-p *ldapport*	Specify the TCP port where the LDAP server is listening. The default LDAP non-secure port is 389 and the default LDAP secure port is 636.
-d *debuglevel*	Set the LDAP debug level. The debug level you specify must be a decimal value. You can also specify the levels additively if you want to specify two or more. For example, specify 32768 to turn on ERROR and specify 1 to turn on TRACE, or specify 32769 (32768+1) to turn on both ERROR and TRACE. See Table 18 on page 102 for the possible decimal values for *debuglevel*.
-D *binddn*	Use *binddn* to bind to the LDAP directory. The *binddn* parameter should be a string-represented DN. The default is a NULL string. If the **-S** or **-m** option is equal to **DIGEST-MD5** or **CRAM-MD5**, this option is the authorization DN which will be used for making access checks. This directive is optional when used in this manner.

-w *bindpasswd*	Use *bindpasswd* as the password for simple authentication. The default is a NULL string.
-b *searchbase*	Use *searchbase* as the starting point for the search instead of the default. If **-b** is not specified, this utility examines the **LDAP_BASEDN** environment variable for a searchbase definition. If you are running in TSO, set the **LDAP_BASEDN** environment variable using Language Environment runtime environment variable **_CEE_ENVFILE**. See *z/OS C/C++ Programming Guide* for more information. If you are running in the z/OS shell, simply export the **LDAP_BASEDN** environment variable.
-Z	Use a secure connection to communicate with the LDAP server. Secure connections expect the communication to begin with the SSL/TLS handshake. The **-K** (*keyfile*) option or equivalent environment variable is required when the **-Z** option is specified. The **-P** (*keyfilepw*) option is required when the **-Z** option is specified and the key file specifies an HFS key database file. The **-N** (*keyfilelabel*) option must be specified if you wish to use a certificate that is different than the default specified in the key database.
-K *keyfile*	Specify the name of the System SSL key database file or RACF key ring. If this option is not specified, this utility looks for the presence of the **SSL_KEYRING** environment variable with an associated name. System SSL assumes that the name specifies a key database file. If the name is not a fully-qualified file name, then the current directory is assumed to contain the file. The key database file must be a file and cannot be an MVS dataset. If a corresponding file is not found then the name is assumed to specify a RACF key ring. See “SSL/TLS information for LDAP utilities” on page 136 for information on System SSL key databases and RACF key rings. This parameter is ignored if **-Z** is not specified.
-P *keyfilepw*	Specify either the key database file password or the file specification for a System SSL password stash file. When the stash file is used, it must be in the form *file://* followed immediately (no blanks) by the HFS file specification (e.g. file:///etc/ldap/sslstashfile). The stash file must be a file and cannot be an MVS dataset. This parameter is ignored if **-Z** is not specified.
-N *keyfiledn*	Specify the certificate name in the key database file.
-U *userName*	Specify the *userName* for **CRAM-MD5** or **DIGEST-MD5** binds. The *userName* is a short name (uid) that will be used to perform bind authentication. This option is required if the **-S** or **-m** option is set to DIGEST-MD5.
-g *realmName*	Specify the *realmName* to use when doing a **DIGEST-MD5** bind. This option is required when multiple realms are passed from an LDAP server to a client as part of a **DIGEST-MD5** challenge; otherwise, it is optional.

filter

Specify an IETF RFC 1558 compliant LDAP search filter. UUID will be added only to those entries that match the filter.

Examples

Following are some **ldapadduuids** examples.

- `ldapadduuids -D "cn=ldap administrator" -w password -b "ou=Home Town,o=ibm_us,c=us" "cn=H*"`

  `GLD2104I ldapadduuids added ibm-entryuuid to 1 entries.`

This example adds UUIDs to all entries starting with "H" in the **cn** attribute in the `ou=Home Town` subtree that do not already have an **ibm-entryuuid** attribute.

- `ldapadduuids -D "cn=ldap administrator" -w password -b "ou=Home Town,o=ibm_us,c=us"`

  `GLD2104I ldapadduuids added ibm-entryuuid to 51 entries.`

This example adds UUIDs to all entries in the `ou=Home Town` subtree that did not already have one.

Usage

1. This utility only works with a TDBM backend.
2. With this utility you can break up the task of adding UUIDs to the database by specifying a subset of the database to add them to. This is useful when the directory database is very large and adding the UUIDs would take more time than your administrator's change window. It is also useful when you know that a portion of the directory tree will not need the entry UUIDs, thus saving space.
3. Note that an **ibm-entryuuid** attribute will not be added to entries that already have one.
4. New entries added to a TDBM database will automatically get an **ibm-entryuuid** attribute.

db2pwden utility

Purpose

The **db2pwden** utility is provided to encrypt all clear text user passwords in an already loaded TDBM backend. The utility runs as a client operation while the server is active, and causes the server to encrypt all the **userPassword** attribute values that are in clear text with the **pwEncryption** method configured on the LDAP server. The utility must be run by the LDAP administrator or a user who has the authority to update passwords.

Format

db2pwden [*options***]**

Parameters

options

The following table shows the *options* you can use for the **db2pwden** utility:

Table 22. db2pwden options

Option	Description
-?	Print this text.
-m *mechanism* or **-S** *mechanism*	Specify the bind method to use. You can use either **-m** or **-S** to indicate the bind method. The default is **SIMPLE**. You can also specify **GSSAPI** to indicate a Kerberos Version 5 is requested, **EXTERNAL** to indicate that a certificate (SASL external) bind is requested, **CRAM-MD5** to indicate that a SASL Challenge Response Authentication Mechanism bind is requested, or **DIGEST-MD5** to indicate a SASL digest hash bind is requested. The **GSSAPI** method requires a protocol level of 3 and the user must have a valid Kerberos Ticket Granting Ticket in their credentials cache by using the Kerberos **kinit** command line utility. The **EXTERNAL** method requires a protocol level of 3. You must also specify **-Z**, **-K**, and **-P** to use certificate bind. If there is more than one certificate in the key database file, use **-N** to specify the certificate or the default certificate will be used. The **CRAM-MD5** method requires protocol level 3. The **-D** or **-U** option must be specified. The **DIGEST-MD5** method requires protocol level 3. The **-U** option must be specified. The **-D** option can optionally be used to specify the authorization DN.
-h *ldaphost*	Specify the host on which the LDAP server is running. The default is the local host. When the target host is a z/OS LDAP server operating in multi-server mode with dynamic workload management enabled (see “Determining operational mode” on page 82 for additional information about LDAP server operating modes), the *ldaphost* value should be in the form *group_name.sysplex_domain_name*, where *group_name* is the name of the **sysplexGroupName** identified in the server configuration file and *sysplex_domain_name* is the name or alias of the sysplex domain in which the target server operates.
-p *ldapport*	Specify the TCP port where the LDAP server is listening. The default LDAP non-secure port is 389 and the default LDAP secure port is 636.
-d *debuglevel*	Specify the level of debug messages to be created. The debug level is specified in the same fashion as the debug level for the LDAP server, as described on page 102. Table 18 on page 102 lists the specific debug levels. The default is no debug messages.
-D *binddn*	Use *binddn* to bind to the LDAP directory. The *binddn* parameter should be a string-represented DN. The default is a NULL string. If the **-S** or **-m** option is equal to **DIGEST-MD5** or **CRAM-MD5**, this option is the authorization DN which will be used for making access checks. This directive is optional when used in this manner.

Table 22. db2pwden options (continued)

Option	Description
-w *bindpasswd*	Use *bindpasswd* as the password for simple authentication. The default is a NULL string.
-b *base*	Use *base* as the starting point for the update instead of the default. If **-b** is not specified, this utility examines the **LDAP_BASEDN** environment variable for a *base* definition. If you are running in TSO, set the **LDAP_BASEDN** environment variable using Language Environment runtime environment variable **_CEE_ENVFILE**. See *z/OS C/C++ Programming Guide* for more information. If you are running in the z/OS shell, export the **LDAP_BASEDN** environment variable.
-Z	Use a secure connection to communicate with the LDAP server. Secure connections expect the communication to begin with the SSL/TLS handshake. The **-K** (*keyfile*) option or equivalent environment variable is required when the **-Z** option is specified. The **-P** (*keyfilepw*) option is required when the **-Z** option is specified and the key file specifies an HFS key database file. The **-N** (*keyfilelabel*) option must be specified if you wish to use a certificate that is different than the default specified in the key database.
-K *keyfile*	Specify the name of the System SSL key database file or RACF key ring. If this option is not specified, this utility looks for the presence of the **SSL_KEYRING** environment variable with an associated name. System SSL assumes that the name specifies a key database file. If the name is not a fully-qualified file name, then the current directory is assumed to contain the file. The key database file must be a file and cannot be an MVS dataset. If a corresponding file is not found then the name is assumed to specify a RACF key ring. See “SSL/TLS information for LDAP utilities” on page 136 for information on System SSL key databases and RACF key rings. This parameter is ignored if **-Z** is not specified.
-P *keyfilepw*	Specify either the key database file password or the file specification for a System SSL password stash file. When the stash file is used, it must be in the form *file://* followed immediately (no blanks) by the HFS file specification (e.g. file:///etc/ldap/sslstashfile). The stash file must be a file and cannot be an MVS dataset. This parameter is ignored if **-Z** is not specified.
-N *keyfiledn*	Specify the label associated with the key in the key database file.
-U *username*	Specify the *userName* for **CRAM-MD5** or **DIGEST-MD5** binds. The *userName* is a short name (uid) that will be used to perform bind authentication. This option is required if the **-S** or **-m** option is set to DIGEST-MD5.
-g *realmname*	Specify the *realmName* to use when doing a **DIGEST-MD5** bind. This option is required when multiple realms are passed from an LDAP server to a client as part of a **DIGEST-MD5** challenge; otherwise, it is optional.

Examples

Following are some **db2pwden** examples:

- The following command:

```
db2pwden -D "cn=admin" -w secret
```

  encrypts all clear text user passwords in the TDBM backend at the LDAP server on the local host. The encryption method used is the **pwEncryption** method configured on the LDAP server.

- The following command:

```
db2pwden -h ushost -p 391 -D "cn=admin" -w secret -b "o=university, c=US"
```

encrypts all clear text user passwords starting at the base "o=university,c=US" in the TDBM backend on host ushost at port 391. The encryption method used is the **pwEncryption** method configured on the LDAP server.

Diagnostics: Exit status is 0 if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error.

Chapter 12. Internationalization support

This chapter discusses translated messages and UTF-8 support.

Translated messages

The **LANG** and **NLSPATH** environment variables are set for the LDAP Server and the LDAP programs in the server's **envvars** file. The default name and location for this file is:

```
/etc/ldap/slapd.envvars
```

There are no default values for these variables. Messages are also available in Japanese. The **LANG** variable should be set to **LANG=Ja_JP** or **Ja_JP.IBM-939**. These variables should also be set either in the environment variable file of the user or by exporting the variables in the shell for the user ID that will run the LDAP utilities.

A sample **slapd.envvars** file for the LDAP server is shipped in **/usr/lpp/ldap/etc**. You can copy this file to **/etc/ldap** and modify its contents. Following is part of the sample file.

```
NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lib/nls/msg/En_US.IBM-1047/%N
LANG=En_US.IBM-1047
```

There are symbolic links to the English language message catalogs in the following locations:

- **/usr/lib/nls/msg/C**
- **/usr/lib/nls/msg/En_US**
- **/usr/lib/nls/msg/En_US.IBM-1047**

It is possible to run with either **LANG=En_US**, **LANG=C**, or **LANG=En_US.IBM-1047** and access the English language LDAP message catalogs.

There are also symbolic links to the following Japanese language message catalogs:

- **/usr/lib/nls/msg/Ja_JP**
- **/usr/lib/nls/msg/Js_JP.IBM-939**

UTF-8 support

UTF stands for "UCS (Unicode) Transformation Format". The UTF-8 encoding can be used to represent any Unicode character. Depending on a Unicode character's numeric value, the corresponding UTF-8 character is a 1, 2, or 3 byte sequence. Table 23 shows the mapping between Unicode and UTF-8. Refer to IETF RFC 2279, See IETF RFC 2253 *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names* for more information on UTF-8.

Table 23. Mapping between Unicode and UTF-8

Unicode range (hexadecimal)	UTF-8 octet sequence (binary)
0000-007F	0xxxxxxx
0080-07FF	110xxxxx 10xxxxxx
0800-FFFF	1110xxxx 10xxxxxx 10xxxxxx

The LDAP Version 3 protocol specifies that all data exchanged between LDAP clients and servers be UTF-8. The z/OS LDAP server supports UTF-8 data exchange as part of its Version 3 protocol support.

Note: For UTF-8 data stored in a z/OS LDAP server's TDBM and GDBM backends, collation for single-byte UTF-8 characters is relative to the server's locale. For multi-byte UTF-8 characters, collation is relative to the numeric value of the equivalent Unicode character.

Part 2. Use

Chapter 13. Data model

The LDAP data model is closely aligned with the X.500 data model. In this model, a directory service provides a hierarchically organized set of *entries*. Each of these entries is represented by an *object class*. The object class of the entry determines the set of *attributes* which are required to be present in the entry as well as the set of attributes that can optionally appear in the entry. An attribute is represented by an *attribute type* and one or more *attribute values*. In addition to the attribute type and values, each attribute has an associated *syntax* which describes the format of the attribute values. Examples of attribute syntaxes for LDAP include directory string and binary.

To summarize, the directory is made up of entries. Each entry contains a set of attributes. These attributes can be single or multi-valued (have one or more values associated with them). The object class of an entry determines the set of attributes that must exist and the set of attributes that may exist in the entry. Refer to *SC24-5906 z/OS, DCE Application Development Guide: Directory Services* for more information on the X.500 Directory Information Model.

Every entry in the directory has a *distinguished name (DN)*. The DN is the name that uniquely identifies an entry in the directory. A DN is made up of attribute=value pairs, separated by commas. For example:

```
cn=Ben Gray,ou=editing,o=New York Times,c=US
cn=Lucille White,ou=editing,o=New York Times,c=US
cn=Tom Brown,ou=reporting,o=New York Times,c=US
```

Any non-binary attributes defined in the directory schema may be used to make up a DN.

Note: The z/OS LDAP server does not allow a DN that contains a **userpassword** attribute.

The order of the component attribute=value pairs is important. The DN contains one component for each level of the directory hierarchy. LDAP DNs begin with the most specific attribute (usually some sort of name), and continue with progressively broader attributes, often ending with a country attribute.

Relative distinguished names

Each component of a DN is referred to as a *relative distinguished name (RDN)*. It identifies an entry distinctly from any other entries which have the same parent. In the examples above, the RDN `cn=Ben Gray` separates the first entry from the second entry, (with RDN `cn=Lucille White`). The attribute=value pair or pairs making up the RDN for an entry must also be present as an attribute=value pair or pairs in the entry. This is not true of the other components of the DN. When using the TDBM backend, TDBM adds the attribute=value pairs in the RDN to the entry if they are not already present.

RDNs can contain multiple attribute=value pairs. So-called multivalued RDNs use two or more attribute=value pairs from the directory entry to define the name of the entry relative to its parent. An example where this would be useful would be where a directory hierarchy of users was being defined for a large university. This hierarchy would be segmented by campus. A problem is encountered, however, when it is discovered that there is more than one John Smith at the downtown campus. The RDN cannot simply be the name of the user. What can be done, however, is to add a unique value to the RDN, thus ensuring its uniqueness across the campus. Typically universities hand out serial numbers to their students. Coupling the student number with the person's name is one method of solving the problem of having a unique RDN under a parent in the directory hierarchy. The entry's RDN might look something like:

```
cn=John Smith+studentNumber=123456.
```

The plus sign (+) is used to delimit separate attribute=value pairs within an RDN. The entry's DN might look like:

```
 cn=John Smith+studentNumber=123456, ou=downtown, o=Big University, c=US
```

Distinguished name syntax

The Distinguished Name (DN) syntax supported by this server is based on IETF RFC 1779 *A String Representation of Distinguished Names* and IETF RFC 2253 *LDAP (v3): UTF-8 String Representation of Distinguished Names*. A semicolon (;) character may be used to separate RDNs in a distinguished name, although the comma (,) character is the typical notation.

White space (blank) characters may be present on either side of the comma or semicolon. The white space characters are ignored, and the semicolon replaced with a comma.

In addition, space characters may be present between an attribute=value pair and a plus sign (+), between an attribute type and an equal sign (=), and between an equal sign (=) and an attribute value. These space characters are ignored when parsing.

A value may be surrounded by quotation marks, which are not part of the value. Inside the quoted value, the following characters can occur without any escaping:

- A space or pound sign (#) character occurring at the beginning of the string
- A space character occurring at the end of the string
- One of the characters
  - apostrophe (')
  - equal sign (=)
  - plus sign (+)
  - backslash (\)
  - less than sign (<)
  - greater than sign (>)
  - semicolon (;)

Alternatively, a single character to be escaped may be prefixed by a backslash (\). This method may be used to escape any of the characters listed above, plus the quotation mark.

This notation is designed to be convenient for common forms of name. This section gives a few examples of distinguished names written using this notation. First is a name containing three components:

```
OU=Sales+CN=J. Smith,O=Widget Inc.,C=US
```

This example shows a method of escaping a comma in an organization name:

```
CN=R. Smith,O=Big Company\, Inc.,C=US
```

Domain component naming

Domain component naming as specified by RFC 2247 is also supported in the LDAP server. For example, the domain name `ibm.com` could be specified as an entry in the LDAP server with the following distiguished name:

```
dc=ibm,dc=com
```

RACF-style distinguished names

If you are using SDBM (the RACF database backend of the LDAP server), the format of the DNs is restricted in order to match the schema of the underlying RACF data. A RACF-style DN for a user or group contains two required attributes plus a suffix:

racfid Specifies the user ID or group ID.

profiletype
Specifies **user** or **group**.

suffix Specifies the SDBM suffix.

A RACF-style DN for a user's connection to a group also contains two required attributes plus a suffix:

racfuserid+racfgroupid
Specifies the user and the group.
profiletype
Specifies **connect**.
suffix Specifies the SDBM suffix.

The suffix for SDBM may contain additional attributes. For example, if the suffix has been specified as:

```
suffix cn=myRACF,c=US
```

in the LDAP configuration file, any RACF-style DN would end with:

```
cn=myRACF,c=US
```

Following is an example of the DN format and a sample DN for a user:

```
racfid=userid,profiletype=user,suffix
racfid=ID1,profiletype=user,cn=myRACF,c=US
```

Following is an example of the DN format and a sample DN for a connection:

```
racfuserid=userid+racfgroupid=groupid,profiletype=connect,suffix
racfuserid=ID1+racfgroupid=GRP1,profiletype=connect,cn=myRACF,c=US
```

Chapter 14. LDAP directory schema

The LDAP Version 3 (V3) protocol, as defined in IETF RFC 2252 *Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions* and IETF RFC 2256 *A Summary of the X.500(96) User Schema for use with LDAPv3*, describes schema publication and update. Schema publication provides the ability to query the active directory schema through the use of the LDAP search function. Schema update is the ability to change the schema while the directory server is running.

Note:

- The z/OS LDAP server implementation of both schema publication and update is provided for the TDBM and GDBM backend.
- When the z/OS LDAP server is first started with TDBM or GDBM configured, the server supplies an initial minimum schema. This initial schema is sufficient for usage of the GDBM backend, but will need to be updated for usage of TDBM.

Schema publication only is available for the SDBM backend.

Setting up the schema for TDBM - new users

The LDAP server is shipped with two predefined schema files representing schema definitions which the user may want to load as the LDAP schema TDBM. For TDBM, the schema is stored as an entry in the database and modify operations may be performed on this entry. These files are **schema.user.ldif** and **schema.IBM.ldif**. The **schema.IBM.ldif** schema definitions require that the definitions contained in **schema.user.ldif** are loaded prior to loading **schema.IBM.ldif**. Determine which schema files would be used to represent the data stored in the TDBM database. Copy the files from the **/usr/lpp/ldap/etc** directory to a working directory, for example, the **/home/myuser** directory. For each file, find the line

```
dn: cn=schema, <suffix>
```

and replace <suffix> with one of the suffixes defined for TDBM in the LDAP server configuration file. In the example below, the suffix defined in the configuration file for TDBM is `o=Your Company,c=US`. Then run the **ldapmodify** command from the z/OS shell specifying the host, port, bind DN, password, and schema file for each schema file to be loaded. This will load the schema into the directory.

For example:

1. Copy the schema file(s) to a working directory:

   ```
   cp /usr/lpp/ldap/etc/schema.user.ldif /home/myuser/schema.user.ldif
   cp /usr/lpp/ldap/etc/schema.IBM.ldif /home/myuser/schema.IBM.ldif
   ```

2. Edit the schema file(s) and replace

   ```
   dn: cn=schema, <suffix>
   ```

   with

   ```
   dn:cn=schema, o=Your Company, c=US
   ```

3. Run the **ldapmodify** command(s):

   ```
   ldapmodify -h ldaphost -p ldapport -D adminDN -w passwd -f /home/myuser/schema.user.ldif
   ldapmodify -h ldaphost -p ldapport -D adminDN -w passwd -f /home/myuser/schema.IBM.ldif
   ```

   See *z/OS Integrated Security Services LDAP Client Programming* for more information about **ldapmodify**.

Upgrading schema for TDBM

The schema files that are shipped with the z/OS LDAP server are based on industry and product defined schemas. As such, they should not be modified since existing products and applications use the schema elements as defined.

Prior to z/OS V1R6, both individual schema files related to specific uses and combined schema files were shipped with the LDAP server to be used for TDBM. In z/OS V1R6, only the combined schema files, **schema.user.ldif** and **schema.IBM.ldif**, are shipped. These files contain all of the contents of the individual schema files, plus some additional schema elements.

Determine which of the following cases applies to your LDAP installation:

1. If the TDBM backend has never been configured and started, refer to "Setting up the schema for TDBM - new users" on page 165 for instructions on loading the schema for TDBM.
2. If you are migrating from an RDBM backend to a TDBM backend, refer to "Migrating the schema from RDBM to TDBM" on page 183.
3. If you are currently running a TDBM backend on a previous release of z/OS LDAP, refer to "Updates to the schema."

Updates to the schema

Occasionally, schema updates are required during the life of an LDAP release. These updates will be applied to and shipped with the **schema.user.ldif** and **schema.IBM.ldif** files found in **/usr/lpp/ldap/etc** directory. When moving to a new release you must re-apply both of these files. Future schema service will depend on those updates being applied to your schema.

When either the **schema.user.ldif** or **schema.IBM.ldif** file is shipped in the service stream or you are moving to a new release, the LDAP Administrator should:

1. Copy the file(s) to a local directory
2. Replace `<suffix>` in the `cn=schema, <suffix>` line in the file(s), using one of the suffixes defined for this backend in the configuration file.
3. Update their TDBM schema through the **ldapmodif**y utility.

Note: Check that **schemaReplaceByValue off** is not specified in the backend section of the configuration file or send the **schemaReplaceByValueControl** control with a value of **TRUE** on the modify request. This control can be sent by specifying the **-u** option on the **ldapmodify** utility. Refer to Chapter 8, "Customizing the LDAP server configuration," on page 53 for more information on the **schemaReplaceByValue** configuration option and to Appendix D, "Supported server controls," on page 459 for more information on the **schemaReplaceByValueControl** control.

Schema introduction

DWL-AG660

Entries in the directory are made up of attributes which consist of an attribute type and one or more attribute values. These are referred to as *attribute=value* pairs. Every entry contains one or more **objectClass** *attribute=value* pairs that identify what type of information the entry contains. The object classes associated with the entry determine the set of attributes which must or may be present in the entry.

The schema is represented and stored as another entry in the directory. Following is a portion of the schema entry.

```
cn=SCHEMA,o=Your Company,c=US
subtreespecification=NULL
objectclass=TOP
objectclass=SUBSCHEMA
objectclass=SUBENTRY
objectclass=IBMSUBSCHEMA
...
attributetypes= ( 2.5.4.3 NAME ( 'cn' 'commonName' ) SUP name )
...
ibmattributetypes = ( 2.5.4.3 ACCESS-CLASS normal )
...
objectclasses = ( 2.5.6.0 NAME 'top' ABSTRACT MUST objectclass )
...
ldapsyntaxes = ( 1.3.6.1.4.1.1466.115.121.1.15 DESC 'directory string' )
...
matchingrules = ( 2.5.13.5 NAME 'caseExactMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
...
```

Figure 16. Sample Schema Entry

The **objectClass** values specified for the schema entry are **top**, **subEntry**, **subSchema**, and **ibmSubschema**. This set of object classes result in the **objectClass**, **cn**, and **subtreeSpecification** attributes being required for a schema entry and the **attributeTypes**, **objectClasses**, **ldapSyntaxes**, **matchingRules**, and **ibmAttributeTypes** attributes being allowed in a schema entry.

Note: The **ditContentRules**, **ditStructureRules**, **nameforms**, and **matchingRuleUse** attributes are allowed in a schema entry, but usage of these directives is not implemented by the z/OS LDAP server.

Every entry in the directory including the schema entry contains the **subschemaSubentry** attribute. The value shown for this attribute is the DN of the schema entry which contains the definitions for the directory entry. For entries stored in TDBM, the **subschemaSubentry** value will be `cn=schema,`*`suffix`* where *`suffix`* is the TDBM suffix under which this directory entry is stored. For example, assuming the suffix is `'o=Acme Company, c=UK'`, for the directory entry

```
cn= Mary Smith, o=Acme Company, c=UK
objectClass= person
cn= Mary Smith
sn= Smith
```

the **subschemaSubentry** value would be

```
subschemaSubentry= cn=schema, o=Acme Company, c=UK
```

Attribute types, object classes, LDAP syntaxes, and matching rules have assigned unique numeric object identifiers. These numeric object identifiers are in dotted decimal format, for example, 2.5.6.6. Attribute types, object classes, and matching rules are also identified by a textual name, for example, `person` or `names`. The numeric object identifier and the textual names may be used interchangeably when an attribute type or object class definition specifies an object identifier. Most schema definitions use the textual name as the object identifier for these definitions.

The attributes that comprise a directory schema include attribute types, IBM attribute types, object classes, LDAP syntaxes, and matching rules. There is a fixed set of LDAP syntaxes and matching rules supported by the z/OS LDAP server. These are listed in Table 27 on page 173, Table 28 on page 174, and Table 29 on page 175. Each of the schema attributes are described below:

- **Attribute types**

Attribute types define the characteristics of the data values stored in the directory. Each attribute type defined in a schema must contain a unique numeric object identifier and optionally contain a textual name, zero or more alias names, and a description of the attribute type. The characteristics defined for each attribute type include the syntax, length, and matching rules.

The **SYNTAX** defines the format of the data stored for the attribute type. The server checks the attribute values that are to be added to the directory by comparing the values against the set of allowed characters based on the syntax. For example, if the syntax of an attribute type is Boolean (where the acceptable values are **TRUE** or **FALSE**) and the attribute value specified is **yes**, the update will fail. The syntaxes supported by the z/OS LDAP server are shown in Table 27 on page 173 and Table 28 on page 174.

Matching rules may be specified for **EQUALITY**, **ORDERING**, and **SUBSTR** (substring matching). The matching rule determines how comparisons between values will be done. The **EQUALITY** matching rule determines if two values are equal. Examples of **EQUALITY** matching rules are **caseIgnoreMatch**, **caseExactMatch**, and **telephoneNumberMatch**. The **ORDERING** matching rule determines how two values are ordered (**greaterThanOrEqual**, **lessThanOrEqual**). Examples of **ORDERING** matching rules are **caseIgnoreOrderingMatch** and **generalizedTimeOrderingMatch**. The **SUBSTR** matching rule determines if the presented value is a substring of an attribute value from the directory. Examples of **SUBSTR** matching rules are **caseIgnoreSubstringsMatch** and **telephoneNumberSubstringsMatch**.

If **EQUALITY**, **ORDERING**, or **SUBSTR** matching rules are not specified in the definition of an attribute type or through the inheritance hierarchy, the z/OS LDAP server will perform evaluations to the best of its ability, but the results may not be as expected. The z/OS LDAP server uses the matching rules shown in the following table based on attribute type syntax to evaluate **EQUALITY** if the **EQUALITY** matching rule is not specified.

Table 24. Syntax and default EQUALITY matching rule

Syntax	Equality
Attribute Type Description	objectIdentifierFirstComponentMatch
Binary	-
Boolean	caseIgnoreMatch
Directory String	caseIgnoreMatch
DIT Content Rule Description	objectIdentifierFirstComponentMatch
DIT Structure Rule Description	integerFirstComponentMatch
DN	distinguishedNameMatch
Generalized Time	generalizedTimeMatch
IA5 String	caseIgnoreIA5Match
IBM Attribute Type Description	objectIdentifierFirstComponentMatch
IBM Entry UUID	IBM-EntryUUIDMatch
Integer	integerMatch
LDAP Syntax Description	objectIdentifierFirstComponentMatch
Matching Rule Description	objectIdentifierFirstComponentMatch
Matching Rule Use Description	objectIdentifierFirstComponentMatch
Name Form Description	objectIdentifierFirstComponentMatch
Object Class Description	objectIdentifierFirstComponentMatch
Object Identifier	caseIgnoreMatch
Octet String	-
Substring Assertion	-
Telephone Number	telephoneNumberMatch
UTC Time	utcTimeMatch

The z/OS LDAP server also verifies that the matching rules specified for **EQUALITY**, **ORDERING**, and **SUBSTR** are consistent with the specified **SYNTAX**. Table 25 shows acceptable values **EQUALITY**, **ORDERING**, and **SUBSTR**.

Table 25. Syntax and acceptable matching rules (EQUALITY, ORDERING, and SUBSTR)

Syntax	EQUALITY	ORDERING	SUBSTR
Attribute Type Description	objectIdentifierFirstComponentMatch	-	-
Binary	-	-	-
Boolean	booleanMatch caseIgnoreMatch caseIgnoreIA5Match caseExactMatch caseExactIA5Match	caseIgnoreOrderingMatch caseExactOrderingMatch	caseIgnoreSubstringsMatch caseExactSubstringsMatch
Directory String	caseIgnoreMatch caseExactMatch	caseIgnoreOrderingMatch caseExactOrderingMatch	caseIgnoreSubstringsMatch caseExactSubstringsMatch
DIT Content Rule Description	objectIdentifierFirstComponentMatch	-	-
DIT Structure Rule Description	integerFirstComponentMatch	-	-
DN	distinguishedNameMatch	distinguishedNameOrdering Match	-
Generalized Time	generalizedTimeMatch	generalizedTimeOrdering Match	-
IA5	caseIgnoreMatch caseIgnoreIA5Match caseExactMatch caseExactIA5Match	caseIgnoreOrderingMatch caseExactOrderingMatch	caseIgnoreSubstringsMatch caseExactSubstringsMatch
IBM Attribute Type Description	objectIdentifierFirstComponent Match	-	-
IBM Entry UUID	IBM-EntryUUIDMatch	-	-
Integer	integerMatch	-	-
LDAP Syntax Description	objectIdentifierFirstComponentMatch	-	-
Matching Rule Description	objectIdentifierFirstComponentMatch	-	-
Matching Rule Use Description	objectIdentifierFirstComponentMatch	-	-
Name Form Description	objectIdentifierFirstComponentMatch	-	-
Object Class Description	objectIdentifierFirstComponentMatch	-	-
Object Identifier	objectIdentifierMatch	-	-
Octet String	octetStringMatch	-	-
Substring Assertion	-	-	-
Telephone Number	telephoneNumberMatch	-	telephoneNumberSubstrings Match
UTC Time	utcTimeMatch	generalizedTimeOrdering Match	-

The syntax or matching rule values may be inherited by specifying a superior attribute type. This is done by specifying the keyword **SUP**, followed by the object identifier of the superior attribute type. This is known as an attribute type hierarchy and referred to as inheritance. A superior hierarchy may be created with multiple levels of inheritance. In the following partial example, `ePersonName` and `personName` would inherit their **SYNTAX** from `name`.

```
ePersonName SUP personName
personName SUP name
name SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
```

When the **SYNTAX**, **EQUALITY**, **ORDERING**, or **SUBSTR** values are not specified for an attribute type, the attribute type hierarchy will be used to determine these values. The **SYNTAX** must be specified on the attribute type or through inheritance.

The number of values that may be stored in each entry for an attribute type is limited to one value if the keyword **SINGLE-VALUE** is specified. Otherwise, any number of attribute values may exist in the entry.

The **OBSOLETE** keyword indicates that the attribute type cannot be used to add data to existing entries or to store data in new entries. Modifications to entries which contain data values of an attribute type which has been made obsolete will fail unless all data values for all obsolete attribute types are removed during the modification. Searches specifying the obsolete attribute type will return the entries containing the attribute type. If an obsolete attribute type is referred to in a superior hierarchy, the inherited values will continue to be resolved.

Example 1:

```
attributeTypes: ( 1.2.3.4 NAME 'obsattr1' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 OBSOLETE )
attributeTypes: ( 5.6.7.8 NAME 'validattr1' SUP obsattr1 )
```

would be the same as

```
attributeTypes: ( 5.6.7.8 NAME 'validattr' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

Example 2:

```
attributeTypes: ( 10.20.30.40 NAME 'obsattr2' SUP obsattr3 )
attributeTypes: ( 50.60.70.80 NAME 'obsattr3'
EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
attributeTypes: ( 90.100.110.120 NAME 'validattr2' SUP obsattr2 )
```

would be the same as

```
attributeTypes: ( 90.100.110.120 NAME 'validattr2'
EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

The **USAGE** keyword's valid values are **userApplications** or one of three operational values (**directoryOperation**, **distributedOperation**, or **dSAOperation**). An attribute type which has an operational **USAGE** value is called an operational attribute. Operational attributes are treated differently than non-operational attributes. In particular, the value of an operational attribute type in an entry is only returned by a search operation if the attribute type is specified in the list of attributes to be returned. Also, operational attribute types do not have to belong to an object class The default for USAGE is **userApplications**.

The z/OS LDAP server restricts users from modifying data values specified for an attribute type when **NO-USER-MODIFICATION** is specified on the definition of the attribute type.
NO-USER-MODIFICATION should only be specified for attribute types that are set by the server because they cannot be assigned a value by the user.

Note: The LDAP V3 protocol also defines a **COLLECTIVE** key word for attribute types. The LDAP server does not support this key word. All attribute types are assumed to be not **COLLECTIVE**.

- **IBM attribute types**

  Additional information required by IBM LDAP servers for each attribute type defined in the schema is specified using the **IBMAttributeTypes** schema attribute. The **IBMAttributeTypes** schema attribute is an extension of the **attributeTypes** schema attribute. If the **attributeTypes** value is not defined, then the corresponding **IBMAttributeTypes** value cannot be defined. For the z/OS LDAP server, the additional information defined using this attribute is the **ACCESS-CLASS** of the associated attribute type.

  The **ACCESS-CLASS** specifies the level of access users have to data values of this attribute type. The levels that may be specified for user-defined attribute types are **normal**, **sensitive**, and **critical**. The **system** and **restricted** keywords are reserved for LDAP server use and are specified for some of the attribute types controlled by the server. See "Attribute access classes" on page 271 for the definition of access classes.

  Note: Other LDAP servers from IBM use the **DBNAME** and **LENGTH** characteristics to specify additional information for their implementations. These may be specified in the schema but are not used by the z/OS LDAP server.

- **Object classes**

Object classes define the characteristics of individual directory entries. The object classes listed in a directory entry determine the set of required and optional attributes for the entry. Each object class defined in a schema must contain a unique numeric object identifier and optionally contain a textual name, zero or more alias names, a description of the object class, and lists of required (**MUST**) or optional (**MAY**) attribute types.

Required and optional attribute types for an object class may be inherited by specifying one or more superior object classes in an object class definition. This is done by specifying the keyword **SUP** followed by the object identifiers of the superior object classes. This is known as an object class hierarchy and referred to as multiple inheritance. A superior hierarchy may be created with multiple levels of inheritance.

Each object class is defined as one of three types: **STRUCTURAL**, **ABSTRACT**, or **AUXILIARY**. The type is specified when the object class is defined. If the type is not specified, it defaults to **STRUCTURAL**.

The structural object class defines the characteristics of a directory entry. Each entry must specify exactly one base structural object class. A base structural object class is defined as the most subordinate object class in an object class hierarchy. **The structural object class of an entry can not be changed.** Once an entry is defined in the directory, it must be deleted and recreated to change the structural object class.

Abstract and auxiliary object classes are used to provide common characteristics to entries with different structural object classes. Abstract object classes are used to derive additional object classes. Abstract object classes must be referred to in a structural or auxiliary superior hierarchy. Auxiliary object classes are used to extend the set of required or optional attribute types of an entry.

When using the keyword **SUP** to create an object class hierarchy, an auxiliary class should only specify superior object classes that are either auxiliary or abstract object classes. Similarly, a structural object class should only specify superior object classes that are either structural or abstract object classes. If these rules are not followed, the z/OS LDAP server might not be able to determine the base structural object class of the entry, resulting in the rejection of the entry.

An example of the relationship between structural, abstract, and auxiliary object classes is the schema entry shown in Figure 16 on page 167. The schema entry specifies **top**, **subEntry**, **subSchema**, and **ibmSubschema** as object classes. The object classes form the following hierarchy:

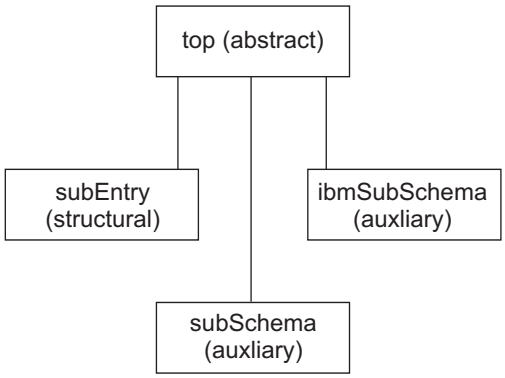


Figure 17. Object class hierarchy example

In this example, the **subEntry** object class is the base structural object class.

The **OBSOLETE** keyword indicates that the object class cannot be used to define entries in the directory. When an object class is made obsolete, new entries specifying the obsoleted object class cannot be added to the directory and existing entries cannot be modified unless the obsolete object class is removed from the entries' object class list. When the obsolete object class is removed from the entry, any attributes in the entry that are associated only with that object class must also be removed.

These changes must be made through the same modify operation. If an obsolete object class is specified in a superior hierarchy for a new entry, then attempts to add the entry to the LDAP directory will fail.

- **LDAP syntaxes**

  Each attribute type definition includes the LDAP syntax which applies to the values for the attribute. The LDAP syntax defines the set of characters which are allowed when entering data into the directory.

  The z/OS LDAP server is shipped with predefined supported syntaxes. See Table 27 on page 173 and Table 28 on page 174 for the list of syntaxes supported by the z/OS LDAP server. The set of syntaxes cannot be changed, added to, or deleted by users.**Matching rules**

  Matching rules allow entries to be selected from the database based on the evaluation of the matching rule assertion. Matching rule assertions are propositions which may evaluate to true, false, or undefined concerning the presence of the attribute value or values in an entry.

  The z/OS LDAP server is shipped with predefined supported matching rules. See Table 29 on page 175 for the list of matching rules supported by the z/OS LDAP server. The set of matching rules cannot be changed, added to, obsoleted, or deleted by users.

Schema attribute syntax

The attributes which are used in the schema entry use specific character representations in their values. These character representations are described in Table 26. The terms shown in this table will be used in the schema attribute definitions in the next section.

Table 26. Character representations

Term	Definition
noidlen	Represented as: *numericoid*{*length*} where *length* is a numeric string representing the maximum length of values of this attribute type. Example: `1.3.6.1.4.1.1466.115.121.1.7{5}` Implementation note: The z/OS LDAP server allows values to be any length, regardless of the specification of a length in the attribute type definition. User installations that want to limit the length of values need to handle this during data input.
numericoid	A dotted decimal string. Example: `2.5.13.72`

oid	A single object identifier. This may be specified either as a name or as a numeric object identifier.
	
	Examples:
	
	`name`
	`2.5.4.41`

oidlist	A list of object identifiers specified as names or numeric object identifiers separated by dollar signs ($) within parentheses. Example: `( cn $ sn $ postaladdress $ 2.5.4.6 )`
oids	Either an *oid* or *oidlist*.

Table 26. Character representations (continued)

Term	Definition

qdescrs	A quoted description shown as '*descr*' for one and as ('*descr*' '*descr*') for more than one. The description (*descr*) must have an alphabetic character as the first character, followed by any combination of alphabetic or numeric characters, the dash character (-), or the semicolon character (;). Each value must be in single quotes (').
	
	If there is more than one value, they must be enclosed in parentheses.
	
	Examples:
	`'x121address'`
	`('cn' 'commonName')`
	`'userCertificate;binary'`
	
	Note: Although the LDAP V3 protocol does not support an underscore character (_) as a valid character in a *descr*, the z/OS LDAP server allows the use of an underscore character to facilitate data migration. This use should be minimized whenever possible and may not be supported by other servers.

qdstring	A quoted descriptive string shown as '*dstring*'. The descriptive string (*dstring*) is composed of one or more UTF-8 characters. Example: `'This is an example of a quoted descriptive string.'`

LDAP schema attributes

The five attributes used to define an LDAP schema are discussed below. For these schema attributes, the *numericoid* must be the first item in the definition. All other keywords and values may be in any order.

LDAP syntaxes

The set of syntaxes which are supported by the z/OS LDAP server cannot be modified, added to, or deleted by users. The descriptive material included here is for information only.

The format of the LDAP syntaxes attribute in a dynamic schema is:

ldapSyntaxes: (*numericoid* [**DESC** *qdstring*] **)**

numericoid
The unique, assigned numeric object identifier.

DESC *qdstring*
Text description of the LDAP syntax

Note: LDAP syntaxes do not have a textual name. They are identified only by the numeric object identifier.

Following is an example of the definition of an LDAP syntax:

```
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.7 DESC 'Boolean' )
```

The LDAP syntaxes supported by the z/OS LDAP server fall into two categories. The first set, as shown in Table 27, would be used when defining attribute types that are used for directory data.

Table 27. Supported LDAP syntaxes - general use

Numeric object identifier	Description	Valid values
1.3.6.1.4.1.1466.115.121.1.5	Binary	Binary data
1.3.6.1.4.1.1466.115.121.1.7	Boolean	TRUE, FALSE
1.3.6.1.4.1.1466.115.121.1.15	Directory String	UTF-8 characters

Table 27. Supported LDAP syntaxes - general use (continued)

Numeric object identifier	Description	Valid values
1.3.6.1.4.1.1466.115.121.1.12	Distinguished Name	Sequence of attribute type and value pairs
1.3.6.1.4.1.1466.115.121.1.24 **Note:** The effective time zone for the LDAP server is assumed when calculating GMT from local time.	Generalized Time	*yyyymmddhhmmss.ffffff* (local time) *yyyymmddhhmmss.ffffff*Z (GMT) *yyyymmddhhmmss.ffffff-diff* (local time with a differential to GMT, where *diff* is *hhmm*)
1.3.6.1.4.1.1466.115.121.1.26	IA5 String	IA5 characters (commonly known as 7-bit ASCII)
1.3.6.1.4.1.1466.115.121.1.27	Integer	+/- 62 digit integer
1.3.6.1.4.1.1466.115.121.1.38	Object Identifier	Name or numeric object identifier
1.3.6.1.4.1.1466.115.121.1.40	Octet String	Octet data
1.3.6.1.4.1.1466.115.121.1.50	Telephone Number	printable string (alphabetic, decimal, ′, **(**, **)**, **+**, **,**, **-**, **.**, **/**, **:**, **?**, and space)
1.3.6.1.4.1.1466.115.121.1.53	UTC Time	See Generalized Time above for details

The second set of syntaxes defined by the z/OS LDAP server are used in the definition of the LDAP schema. These would not typically be used in user schema attribute type definitions. They are listed here for reference.

Table 28. Supported LDAP syntaxes - server use

Numeric object identifier	Description
1.3.6.1.4.1.1466.115.121.1.3	Attribute Type Description
1.3.6.1.4.1.1466.115.121.1.16	DIT Content Rule Description
1.3.6.1.4.1.1466.115.121.1.17	DIT Structure Rule Description
1.3.18.0.2.8.1	IBM Attribute Type Description
1.3.18.0.2.8.3	IBM Entry UUID Description
1.3.6.1.4.1.1466.115.121.1.54	LDAP Syntax Description
1.3.6.1.4.1.1466.115.121.1.30	Matching Rule Description
1.3.6.1.4.1.1466.115.121.1.31	Matching Rule Use Description
1.3.6.1.4.1.1466.115.121.1.35	Name Form Description
1.3.6.1.4.1.1466.115.121.1.37	Object Class Description
1.3.6.1.4.1.1466.115.121.1.58	Substring Assertion

Matching rules

The set of matching rules which are supported by the z/OS LDAP server cannot be modified, added to, obsoleted, or deleted by users. The descriptive material included here is for information only.

The format of the matching rules attribute in a dynamic schema is:

matchingRules: (*numericoid* [**NAME** *qdescrs*] [**DESC** *qdstring*] [**OBSOLETE**] **SYNTAX** *numericoid* **)**

numericoid
The unique, assigned numeric object identifier.

NAME *qdescrs*
The name by which this matching rule is known.

DESC *qdstring*
Text description of the matching rule.

OBSOLETE
Indicates that the matching rule is obsolete.

SYNTAX *numericoid*
Specifies the numeric object identifier of the syntax for this matching rule.

Following is an example of the definition of a matching rule:

```
matchingRules: ( 2.5.13.5 NAME 'caseExactMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

The matching rules supported by the z/OS LDAP server is a fixed set as listed in the following table.

Table 29. Supported matching rules

Numeric object identifier	Name	Syntax
2.5.13.13	booleanMatch	1.3.6.1.4.1.1466.115.121.1.7 (Boolean)
1.3.6.1.4.1.1466.109.114.1	caseExactIA5Match	1.3.6.1.4.1.1466.115.121.1.26 (IA5 String)
2.5.13.5	caseExactMatch	1.3.6.1.4.1.1466.115.121.1.15 (Directory String)
2.5.13.6	caseExactOrderingMatch	1.3.6.1.4.1.1466.115.121.1.15 (Directory String)
2.5.13.7	caseExactSubstringsMatch	1.3.6.1.4.1.1466.115.121.1.58 (Substring Assertion)
1.3.6.1.4.1.1466.109.114.2	caseIgnoreIA5Match	1.3.6.1.4.1.1466.115.121.1.26 (IA5 String)
2.5.13.2	caseIgnoreMatch	1.3.6.1.4.1.1466.115.121.1.15 (Directory String)
2.5.13.3	caseIgnoreOrderingMatch	1.3.6.1.4.1.1466.115.121.1.15 (Directory String)
2.5.13.4	caseIgnoreSubstringsMatch	1.3.6.1.4.1.1466.115.121.1.15 (Directory String)
2.5.13.1	distinguishedNameMatch	1.3.6.1.4.1.1466.115.121.1.12 (Distinguished Name)
1.3.18.0.2.4.405	distinguishedNameOrderingMatch	1.3.6.1.4.1.1466.115.121.1.12 (Distinguished Name)
2.5.13.27	generalizedTimeMatch	1.3.6.1.4.1.1466.115.121.1.24 (Generalized Time)
2.5.13.28	generalizedTimeOrderingMatch	1.3.6.1.4.1.1466.115.121.1.24 (Generalized Time)
1.3.18.0.2.22.2	IBM-EntryUUIDMatch	1.3.18.0.2.8.3 (IBM Entry UUID)
2.5.13.29	integerFirstComponentMatch	1.3.6.1.4.1.1466.115.121.1.27 (Integer)
2.5.13.14	integerMatch	1.3.6.1.4.1.1466.115.121.1.27 (Integer)
2.5.13.0	objectIdentifierMatch	1.3.6.1.4.1.1466.115.121.1.38 (Object Identifier)
2.5.13.30	objectIdentifierFirstComponentMatch	1.3.6.1.4.1.1466.115.121.1.38 (Object Identifier)

Table 29. Supported matching rules (continued)

Numeric object identifier	Name	Syntax
2.5.13.17	octetStringMatch	1.3.6.1.4.1.1466.115.121.1.40 (Octet String)
2.5.13.20	telephoneNumberMatch	1.3.6.1.4.1.1466.115.121.1.50 (Telephone Number)
2.5.13.21	telephoneNumberSubstringsMatch	1.3.6.1.4.1.1466.115.121.1.58 (Substring Assertion)
2.5.13.25	utcTimeMatch	1.3.6.1.4.1.1466.115.121.1.53 (UTC Time)

Attribute types

The format of the attribute types attribute in a dynamic schema is:

```
attributeTypes: ( numericoid  [NAME qdescrs] [DESC qdstring] [OBSOLETE] [SUP oid] [EQUALITY oid]
  [ORDERING oid] [SUBSTR oid] [SYNTAX noidlen] [SINGLE-VALUE] [NO-USER-MODIFICATION]
[USAGE attributeUsage] )
```

numericoid
The unique, assigned numeric object identifier.

NAME *qdescrs*
The name and alias names by which this attribute type is known. This is also known as the object identifier. The first name in the list is used as the base name and the other names are referred to as alias names. It is suggested the shortest name be listed first. If a name is not specified, the numeric object identifier is used to refer to the attribute type.

DESC *qdstring*
Text description of the attribute type.

OBSOLETE
Indicates that the attribute type is obsolete.

SUP *oid*
Specifies the superior attribute type. When a superior attribute type is defined, the **EQUALITY**, **ORDERING**, **SUBSTR**, and **SYNTAX** values may be inherited from the superior attribute type. The referenced superior attribute type must also be defined in the schema. When the **SYNTAX**, **EQUALITY**, **ORDERING**, or **SUBSTR** values are not specified for an attribute type, the attribute type hierarchy will be used to determine these values. The **SYNTAX** must be specified on the attribute type or through inheritance.

EQUALITY *oid*
Specifies the object identifier of the matching rule which is used to determine the equality of values.

ORDERING *oid*
Specifies the object identifier of the matching rule which is used to determine the order of values.

SUBSTR *oid*
Specifies the object identifier of the matching rule which is used to determine substring matches of values.

SYNTAX *noidlen*
The syntax defines the format of the data stored for this attribute type. It is specified using the numeric object identifier of the LDAP syntax and, optionally, the maximum length of data stored for this attribute type.

Implementation note: The z/OS LDAP server allows values to be any length, regardless of the specification of a length in the attribute type definition. User installations that want to manage the lengths of values need to handle this when values are put into the directory.

SINGLE-VALUE
Limits entries to only one value for this attribute type.

NO-USER-MODIFICATION
When specified, users may not modify values of this attribute type.

USAGE *attributeUsage*
Specify **userApplications** for *attributeUsage*. If **USAGE** is not specified, the default is **userApplications**.

The **directoryOperation**, **distributedOperation**, and **DSAOperation** keywords are used to create operational attributes. Operational attributes are treated differently than non-operational attributes. In particular, the value of an operational attribute type in an entry is only returned by a search operation if the attribute type is specified in the list of attributes to be returned. Also, operational attribute types do not have to belong to an object class.

Following are examples of the definition of attribute types:

```
attributeTypes: ( 2.5.4.6 NAME 'c' SUP name SINGLE-VALUE )
attributeTypes: ( 2.5.4.41 NAME 'name' EQUALITY caseIgnoreMatch SUBSTR
   caseIgnoreSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )
```

IBM attribute types

The format of the IBM attribute types attribute in a dynamic schema is:

ibmAttributeTypes: (*numericoid* [**ACCESS-CLASS** *ibmAccessClass*] **)**

numericoid
The unique, assigned numeric object identifier of the associated attribute type.

ACCESS-CLASS *ibmAccessClass*
The level of sensitivity of the data values for this attribute type. The acceptable values are **normal**, **sensitive**, and **critical**. See “Attribute access classes” on page 271 for the definition of these values.

The **IBMAttributeTypes** schema element is an extension of the **attributeTypes** schema element. If the **attributeTypes** value is not defined, then the corresponding **IBMAttributeTypes** value cannot be defined.

Some schema elements are shipped with **ACCESS-CLASS** set to **restricted** or **system**. These values are reserved for LDAP server use. Other IBM LDAP servers may also specify **DBNAME** and **LENGTH** keywords and values. These keywords are not used by the z/OS LDAP server and do not need to be specified when creating schemas. If they are specified in a schema used by the z/OS LDAP server, they will be ignored.

Following is an example of the definition of an IBM attribute type:

```
ibmAttributeTypes: (2.5.4.6 ACCESS-CLASS normal)
```

Object classes

The format of the object classes attribute in a dynamic schema is:

objectClasses: (*numericoid* [**NAME** *qdescrs*] [**DESC** *qdstring*]
[**OBSOLETE**] [**SUP** *oids*] [**ABSTRACT**|**STRUCTURAL**|**AUXILIARY**] {[**MUST** *oids*] [**MAY** *oids*]} **)**

numericoid
The unique, assigned numeric object identifier.

NAME *qdescrs*
The name and alias names by which this object class is known. This is also known as the object identifier. The first name in the list is used as the base name. If name is not specified, the numeric object identifier is used to refer to the object class.

DESC *qdstring*
Text description of the object class.

OBSOLETE
Indicates that the object class is obsolete.

SUP *oids*
List of one or more superior object classes. When a superior object class is defined, entries specifying the object class must adhere to the superset of **MUST** and **MAY** values. The supersets of **MUST** and **MAY** values include all **MUST** and **MAY** values specified in the object class definition and all **MUST** and **MAY** values specified in the object class's superior hierarchy. When an attribute type is specified as a **MUST** in an object class in the hierarchy and a **MAY** in another object class in the hierarchy, the attribute type is treated as a **MUST**. Referenced superior object classes must be defined in the schema.

ABSTRACT | STRUCTURAL | AUXILIARY
Indicates the type of object class. **STRUCTURAL** is the default.

MUST *oids*
List of one or more mandatory attribute types. Attribute types which are mandatory must be specified when adding or modifying a directory entry.

MAY *oids*
List of one or more optional attribute types. Attribute types which are optional may be specified when adding or modifying a directory entry.

The **extensibleObject** object class is an **AUXILIARY** object class which allows an entry to optionally hold any attribute type. The **extensibleObject** object class is supported by the z/OS LDAP server. This allows any attribute type that is known by the schema to be specified in an entry which includes **extensibleObject** in its list of object classes.

The **top** object class is an abstract object class used as a superclass of all structural object classes. For each structural object class, **top** must appear in the **SUP** list of this object class or of an object class in the superior hierarchy of this object class.

Following is an example of the definition of an object class:

```
objectClasses: ( 2.5.6.0 NAME 'top' ABSTRACT MUST objectclass )
objectClasses: ( 2.5.6.6 NAME 'person' SUP top STRUCTURAL MUST ( cn $ sn )
  MAY ( userpassword $ telephonenumber $ seealso $ description ) )
objectClasses: ( 5.6.7.8 NAME 'company' SUP top MUST ( department $ telephoneNumber ) MAY ( postalAddress $ street ) )
objectClasses: ( 1.2.3.4 NAME 'companyPerson' SUP ( company $ person ) )
```

Defining new schema elements

You can define new schema elements for use by applications that you develop to use the directory. You can add new object classes and attribute types to the schema defined to the TDBM and GDBM backends in the server. To define a new object class or attribute type, create an LDIF file containing the new schema information, and perform an LDAP modify operation on the schema entry for the TDBM or GDBM backend. Object classes and attribute types must be defined using the formats described in the previous section, and must include unique numeric object identifiers and names. Ensuring that the numeric object identifier and names are unique is essential to the correct operation of the directory when using your newly defined schema elements.

Numeric object identifiers (OIDs) are strings of numbers, separated by periods. OID "ranges" or "arcs" are allocated by naming authorities. If you are going to define new schema elements, you should obtain an "OID arc" from a naming authority. One such location to get an "OID arc" assigned is managed by Internet Assigned Numbers Authority (IANA) and, can be found at:

```
http://www.iana.org
```

Select the "Application Forms" link and then the "Private Enterprise Number" link to apply for a Private Enterprise number.

Once you have obtained an “OID arc” you can begin assigning OIDs to object classes and attribute types that you define.

For the example below, assume that we have been assigned OID arc `1.3.18.0.2.1000.100`. (**Note**: Do not use this OID arc for defining your own schema elements. This arc is assigned to IBM for its use.) The following example adds a new object class that refers to two new attribute types. As you can see, the object class and attribute types can be added to the schema using a single LDAP modify operation. The changes to the schema are represented in LDIF mode input below:

```
dn: cn=schema, o=Your Company
changetype: modify
add: attributetypes
attributetypes: ( 1.3.18.0.2.1000.100.4.1 NAME 'YourCompanyDeptNo'
   DESC 'A users department number.'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 EQUALITY caseIgnoreMatch
   USAGE userApplications
  )
attributetypes: ( 1.3.18.0.2.1000.100.4.2 NAME 'YourCompanyEmployeeID'
   DESC 'A user employee ID.'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 EQUALITY caseIgnoreMatch
   USAGE userApplications
  )
-
add: objectclasses
objectclasses: ( 1.3.18.0.2.1000.100.6.1 NAME 'YourCompanyPerson'
   DESC 'Attached to inetOrgPerson to add more attributes.'
   SUP top
   AUXILIARY
   MAY ( YourCompanyDeptNo $ YourCompanyEmployeeID )
  )
-
```

The example above assumes that the suffix for the TDBM backend is `o=Your Company`. When you define your own schema, you should set the name of the schema (the “dn:” portion of the LDIF) to the name of the schema entry in the TDBM backend.

This short description has described how to update the schema in the TDBM backend with new schema elements. Defining new schema elements is a complex undertaking and requires a thorough understanding of schema

Updating the schema

Attention

Updating the schema, if not done properly, can result in being unable to access data. Read this section thoroughly to avoid this situation.

When the z/OS LDAP server is first started, a minimum schema is initialized for each backend that is configured. The GDBM and SDBM minimum schema are sufficient for usage of those backends, but the TDBM minimum schema needs to be updated to allow full use of the TDBM backend. The TDBM minimum schema just defines the schema elements which are required by the z/OS LDAP server to read a schema entry. The minimum schema are shown in LDAP search results format in Appendix A, “Minimum schema for TDBM and GDBM,” on page 439 and Appendix B, “SDBM schema,” on page 443. This schema is stored as a directory entry with `dn: cn=schema,`*`suffix`*. After the z/OS LDAP server is first started, the schema attribute types and object classes needed for the TDBM directory should be added to the schema entry using the LDAP modify function. For example, to add the schema contained in the **schema.user.ldif** file, copy the file to a working directory, update the suffix in the file, and specify the file in the **ldapmodify** command as follows:

```
ldapmodify -h ldaphost -p ldapport -D adminDN -w passwd -f /home/myuser/schema.user.ldif
```

See “Setting up the schema for TDBM - new users” on page 165 for more information.

See *z/OS Integrated Security Services LDAP Client Programming* for information on the LDAP utilities.

The operations that can be performed include adding, replacing, or deleting any object class or attribute type that is not part of the minimum schema definition required by the LDAP server. Syntaxes and matching rules cannot be added or removed.

The modifications (additions, changes, and deletions) specified by the LDAP modify function are applied to the schema entry which is retrieved from the database by the server. The resulting schema entry becomes the active schema and is used by the TDBM backend to verify that directory changes adhere to it. This schema entry is then stored back into the directory database.

Updates to the schema must be performed such that the schema fully resolves. This includes:

- All attribute types referred to in object classes must exist in the schema.
- All superior attribute types or object classes must exist.
- Only the syntaxes and matching rules supported by TDBM may be specified in attribute type definitions.
- All attribute types referred to in IBM attribute type definitions must also be defined as attribute types.

The schema may also be changed when using the **ldif2tdbm** utility to load entries into the database. See “ldif2tdbm utility” on page 138 for more information.

Data entries in the directory must adhere to the active schema. If the schema is changed such that an object class has a new **must** attribute type, then whenever an entry which is of that object class is added or modified, the new attribute type must be contained in the entry.

The schema entry is required and, therefore, attempts to delete the schema entry will fail.

The TDBM and GDBM backends will manage dynamic schema changes relative to the sysplex (multiple LDAP servers operating against the same database and multiple threads operating against the database). Changes to schema affect all z/OS LDAP servers operating against the same DB2 tables in multi-server mode.

Note: If you delete schema elements from the active schema when information in the directory still exists which refers to those schema elements, you will lose access to that information. To avoid this situation, modify the schema elements you wish to “delete” by marking them as **OBSOLETE** rather than deleting their definitions from the schema entry. In this way, no new entries can be created using these schema elements and the existing entries which do use these schema elements will still be accessible. Existing entries that use **OBSOLETE** schema elements must be modified to use only non-**OBSOLETE** schema elements during the next modification of the entry in order for the modification to succeed.

Replacing individual schema values

It is often necessary to apply an updated schema file to an existing schema. Optimally, this would replace changed values in the existing schema with their updated values from the file and add new values from the file to the existing schema, leaving all other values in the existing schema unchanged. However, this is not the way the RFC 2251 definition for such a modify with replace operation works: the RFC requires that ALL the existing values in the schema be replaced by the values specified in the schema file. Thus the schema file would have to contain all the unchanged values from the schema in addition to the updated and new values so that no unchanged existing values are lost.

To address this problem, the LDAP server supports two different behaviors when using a modify with replace operation on the schema entry:

1. standard RFC behavior, in which all the existing values for an attribute are replaced by the ones specified in the modify operation
2. schema-replace-by-value behavior, in which each replace value in the modify operation either replaces the existing value (if one exists) in the schema or is added to the schema (if an existing value does not exist). All other values in the schema remain as they are. A replace value replaces a schema value if the schema value and replace value have the same Numeric Object Identifier (NOID). Otherwise, the replace value is considered a new value and is added to the existing values in the schema.

The behavior used by the LDAP server is selected in one of two ways:

1. Specify the **schemaReplaceByValue** configuration option in the TDBM or GDBM backend section of the configuration file to set the behavior for all modify with replace operations of that backend's schema. Specifying **on** activates the schema-replace-by-value behavior; **off** activates the standard RFC behavior. Refer to Chapter 8, "Customizing the LDAP server configuration," on page 53 for more information.
2. Specify the **schemaReplaceByValueControl** control on the modify with replace operation to set the behavior for just that specific modify operation, overriding the **schemaReplaceByValue** configuration option. Specifying **TRUE** in the control activates the schema-replace-by-value behavior; **FALSE** activates the standard RFC behavior. Refer to Appendix D, "Supported server controls," on page 459 for more information.

If neither the **schemaReplaceByValue** configuration option nor the **schemaReplaceByValueControl** control is specified, the default behavior is schema-replace-by-value.

Example: assume that the objectclasses attribute for cn=schema,c=us contains the following values:

```
objectclasses: ( 1.130.255 NAME 'oldObjectclass1' DESC 'old description 1' ... )
objectclasses: ( 1.130.256 NAME 'oldObjectclass2' DESC 'old description 2' ... )
objectclasses: ( 1.130.257 NAME 'oldObjectclass3' DESC 'old description 3' ... )
```

We would like to replace `'oldObjectclass1'` and add a value for `'newObjectclass4'`.

Here is the update file for the modify operation:

```
dn: cn=schema,c=us
changetype: modify
replace: objectclasses
objectclasses: ( 1.130.255 NAME 'newObjectClass1' DESC 'new description 1' ... )
objectclasses: ( 1.3.5.9 NAME 'newObjectClass4' DESC 'description 4' ... )
```

After the modify operation with schema-replace-by-value behavior, the objectclasses attribute in the schema would have the following values:

```
objectclasses: ( 1.130.255 NAME 'newObjectClass1' DESC 'new description 1' ... )
objectclasses: ( 1.130.256 NAME 'oldObjectclass2' DESC 'old description 2' ... )
objectclasses: ( 1.130.257 NAME 'oldObjectclass3' DESC 'old description 3' ... )
objectclasses: ( 1.3.5.9 NAME 'newObjectClass4' DESC 'description 4' ... )
```

If the modify operation with traditional RFC behavior is performed instead, the objectclasses attribute in the schema would end up with the following values:

```
objectclasses: ( 1.130.255 NAME 'newObjectClass1' DESC 'new description 1' ... )
objectclasses: ( 1.3.5.9 NAME 'newObjectClass4' DESC 'description 4' ... )
```

Updating a numeric object identifier (NOID)

It may become necessary to update the Numeric Object Identifier (NOID) of a value in the schema. This NOID change can be accomplished by a special modify operation. The modify operation must consist only of a value to delete followed by a value to add. The value to delete must specify the current NOID of the entry whose NOID is to be changed; the value to add must specify the new NOID for the entry, along with all the other parts of the value. The NAME must be identical in the value to delete and the value to add.

Example: suppose we want to change the NOID of the xyz attribute type from 1.3.5.7 to 2.4.6.8. The update file for the modify operation to accomplish this would look like:

```
cn=schema,c=us
-attributetypes=( 1.3.5.7 NAME 'xyz' DESC 'xyz attribute added for application abc' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 USAGE userApplications )
+attributetypes=( 2.4.6.8 NAME 'xyz' DESC 'xyz attribute added for application abc' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 USAGE userApplications )
```

Changing a NOID should not need to be done as part of normal LDAP server operations. It is intended to be used as an error recovery device for when an incorrect NOID has been added to the schema.

Analyzing schema errors

Following is some information about the possible cause of some schema errors that may be encountered when updating schema:

- For enhanced readability, *type:value* pairs in LDIF files may be split across multiple lines. The indicator to LDIF that the subsequent lines are continuations is that the first character on the subsequent line is a space. This character is ignored by parsers and it is assumed that the next character immediately follows the previous line. Therefore, if a space is needed between the last value on one line and the first value on the subsequent line, a second space needs to exist on the subsequent LDIF line. Various reason codes related to unrecognized values may be issued.
- Attempts to delete or obsolete values that are in the minimum schema will fail. Changes to the minimum schema are not allowed and are ignored by the server.
- The IBM attribute type schema attribute is an extension to the associated attribute type in the schema. If the schema contains an IBM attribute type value for which an attribute type value is not defined, the schema update will fail. For example,

  `ibmAttributeTypes: ( 1.2.3.4 ACCESS-CLASS normal )`

  cannot be specified unless

  `attributeTypes: ( 1.2.3.4 NAME 'sample' ... )`

  is also defined.
- While the UTC Time syntax is supported, usage of the Generalized Time syntax is recommended. For UTC Time syntax, year values between 70 and 99 assume 1970 to 1999 and values between 00 and 69 assume 2000 to 2069.
- When searching attribute type values of GMT or UTC Time syntax, use GMT syntax in the search filter rather than local time. All time values are stored in the data store as GMT times.

Retrieving the schema

The following sections describe how you can display the schema entry and also find the **subschemaSubentry** DN.

Displaying the schema entry

The following command shows how to search for the schema entry:

```
ldapsearch -h ldaphost -p ldapport -s base -b "cn=schema,suffix" "objectclass=subschema"
```

Replace *suffix* with the DN of a TDBM or SDBM suffix from the configuration file, depending on which schema you want to publish or specify `cn=changelog` as the suffix to publish the GDBM schema.

Immediately after the server is started for the first time, this command produces the results shown in Appendix A, “Minimum schema for TDBM and GDBM,” on page 439 when the *suffix* specified is managed by TDBM or GDBM. After the schema has been updated by the administrator, the search results will show the full schema as the union of the minimum schema and the added schema elements. If the *suffix* specified in **ldapsearch** is managed by SDBM, the SDBM schema shown in Appendix B, “SDBM schema,” on page 443 is displayed. The SDBM schema is not modifiable.

The search results when searching the schema entry will show the distinguished name of the schema exactly as entered in the search request. For example:

```
ldapsearch -h ldaphost -p ldapport -s base -b "cn=SCHEMA,o=Acme Company,c=US" "objectclass=subschema"
```

produces:

```
cn=SCHEMA,o=Acme Company,c=US
cn=schema
subtreespecification=NULL
objectclass=TOP
objectclass=SUBSCHEMA
objectclass=SUBENTRY
objectclass=IBMSUBSCHEMA
...
attributetypes = ( 2.5.4.3 NAME ( 'cn' 'commonName' ) SUP name )
...
ibmattributetypes = ( 2.5.4.3 ACCESS-CLASS normal )
...
objectclasses = ( 2.5.6.0 NAME 'top' ABSTRACT MUST objectclass )
...
ldapsyntaxes = ( 1.3.6.1.4.1.1466.115.121.1.15 DESC 'directory string' )
...
matchingrules = ( 2.5.13.5 NAME 'caseExactMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
...
```

Finding the subschemaSubentry DN

The **subschemaSubentry** attribute in each directory entry contains the DN of the schema that was active when the entry was added to the directory. To find the value of the **subschemaSubentry** attribute, specify **subschemaSubentry** as an attribute on an LDAP search of the entry.

```
ldapsearch -h ldaphost -p ldapport -s base -b "o=Acme Company, c=UK" "objectclass=*" subschemasubentry

o=Acme Company, c=UK
subschemasubentry= cn=SCHEMA, o=ACME COMPANY, C=UK
```

Migrating the schema from RDBM to TDBM

z/OS LDAP users who are migrating directories from the RDBM backend to the TDBM backend may also need to migrate the directory schema. The schema would need to be migrated if there had been any changes made to the previously shipped schema files or schema definitions that were added to the schema. The **schema2ldif** tool assists with this conversion. This should be done by the person who understands the schema in place for the directory.

IBM supplied schema

The **schema.user.ldif** and **schema.IBM.ldif** files are the LDAP V3 protocol versions of the **schema.system.at**, **schema.system.oc**, **schema.user.at**, **schema.user.oc**, **schema.IBM.at**, and **schema.IBM.oc** files that were used in previous releases of the z/OS LDAP server for RDBM. All schema elements contained in the listed ***.at** and ***.oc** files are contained either in **schema.user.ldif** or **schema.IBM.ldif**. The **schema.user.ldif** and **schema.IBM.ldif** files are used with the TDBM backend. The ***.ldif** files are different from the ***.at** and ***.oc** files in the following ways:

- LDIF format versus configuration file format.
- Defined in terms of the LDAP Version 3 protocol (IETF RFC 2252 format).
- Use of new syntaxes and separate specification of matching rules.

  The syntaxes supported by RDBM in previous LDAP servers (**bin**, **ces**, **cis**, **dn**, **tel**) have been converted to a combination of a syntax and matching rule. These changes are consistent with the LDAP Version 3 protocol definitions of the attribute types.
- Addition of new attribute types and object classes.
- Use of superior attribute types and object classes in definitions.

For example, the definition of **cn** specifies the attribute type **name** as a superior.

The **schema.IBM.ldif** file requires that the **schema.user.ldif** file be loaded first.

If there were no customer-specific changes or additions made to the previously shipped schema, then the **schema.user.ldif** and **schema.IBM.ldif** files that are shipped with the z/OS LDAP server should be used to load the schema when migrating the directory.

User-defined schema

To minimize the migration effort, it is recommended that customer-specific defined attribute types and object classes be placed in separate **.at** and **.oc** files prior to migration. The output of running the **schema2ldif** utility with these input files would be a separate LDIF file containing the customer-specific schema. This would then be used in conjunction with the z/OS LDAP *.ldif schema files located in **/usr/lpp/ldap/etc**.

The schema2ldif utility

The **schema2ldif** utility takes RDBM and SDBM attribute type and object class files and converts them to the standard LDIF format needed for TDBM.

The **schema2ldif** utility is Java based, so a jar and script file are provided with the z/OS LDAP server. The jar file, named **servertools.jar**, and the script file, named **schema2ldif**, are located in the **/usr/lpp/ldap/sbin** directory. The script file provides an easy manner to run the utility.

This utility can only be run from Unix System Services (USS).

Before running the **schema2ldif** utility, the **PATH** and **CLASSPATH** environment variables in the **schema2ldif** script file should be uncommented and updated if necessary. The export statements listed below are commented out in the **schema2ldif** script file.

- `export CLASSPATH=/usr/lpp/ldap/sbin/servertools.jar:$CLASSPATH`
- `export PATH=/usr/lpp/java/J1.3/bin:$PATH`

Check with your system administrator to find the proper location of Java on your system and adjust the **PATH** and **CLASSPATH** appropriately in the **schema2ldif** script file.

If there are **include** statements in any of your files, those additional files will be processed after the input schema files specifically entered on the command line are parsed and formatted in LDIF format.

Note: The **schema2ldif** utility will add **SUP top** to all object class definitions it encounters.

Format

schema2ldif -s *schemaInputFile1* [*schemaInputFile2...*] **-d** *ldifOutputFile* **-l** *logFile* [**-f** *overrideFile*] [**-o** *OIDFile*] [**-v**] [**-?**]

Parameters

-? Specifies help text for the **schema2ldif** utility.

-s *schemaInputFile1* **[***schemaInputFile2...***]**
Specifies the name of one or more input files to be converted. These files may be attribute type, object class, or **slapd.conf** files from previous releases of the LDAP server. If attribute types and object classes are mixed together in one input schema file, they must be separated into two different files in order for **schema2ldif** to properly parse the file to LDIF.

Following is an example of an attribute type file:

```
attribute   name                        ces   name             50   normal
attribute   acceleratorCapabilities    cis   accelCap         11   normal
attribute   accessHint                 dn    accessHint     1000   normal
attribute   accountHint                dn    accountHint    1000   normal
attribute   accountService             dn    accountService 1000   normal
attribute   AccountSuffix              dn    AccountSuffix  1000   normal
attribute   actionPending              cis   actionPending     5   normal
```

Following is an example of an object class file:

```
objectclass   cacheObject
     requires
             objectClass
     allows
             ttl

objectclass   cimBIOSelement
     requires
             objectClass
     allows
             primaryBIOS
```

Following is a portion of an LDAP configuration file that has server parameters and includes for other files which are the actual attribute and object class files. In this case, the **oc.conf** and **at.conf** files will be opened and parsed to the output LDIF file.

```
include  oc.conf
include  at.conf
sslKeyRingFile  key.kdb
sslKeyRingFilePW  key.kdb
sslCipherSpecs  12288
```

-d *ldifOutputFile*

Specifies the name of the LDIF output file. If there is already a file name the same as the output file specified on the command line, the original file name is renamed to:

ldifOutPutFile.1

All of the output from either the attribute types or the object classes input files are routed to one output file concatenated together.

-l *logFile*

Specifies the name of the log file. After each invocation of the **schema2ldif** utility, a log file is created. If this file is empty, **schema2ldif** completed successfully without errors. Otherwise, errors are printed to this file and to standard output.

-f *overrideFile*

Specifies an optional override file. The purpose of this file is to modify attribute types and object classes from previous releases of the z/OS LDAP server that are encountered in the input schema files. It is strongly recommended that the **system.override** file, shipped with the z/OS LDAP server and located in **/usr/lpp/ldap/etc**, be used.

The format of the override file is:

{*attributeType_Name*|*objectClass_Name*} {**ADD**|**DELETE**|**MODIFY**} [*new_attributeType*|*new_objectClass*]

{attributeType_Name | objectClass_Name}
Name of the attribute type or object class that is being operated on.

ADD | DELETE | MODIFY
Operation that the **schema2ldif** utility will perform to the attribute type or object class.

new_attributeType | new_objectClass
Definition of the attribute type or object class as defined in previous versions of the LDAP server. This field is used when the operation is **ADD** or **MODIFY**.

DELETE
Does not put the specified attribute type or object class definition into the output LDIF file.

ADD
Adds new attribute types or object classes to the output LDIF schema file. For an **ADD** operation of an attribute type, the entire definition must be specified on one line. To add an object class, there must be four arguments specified on the first line: the object class name, **ADD**, the word **objectclass**, and the new object class name. This is followed on subsequent lines with the keyword **requires** or **allows**, and the lists of required or allowed attribute type names.

An attribute type or object class will be added to the output LDIF schema file even if it already exists in one of the schema input files. However, modifying the schema using the output LDIF schema file will fail because of the duplicate attribute type or object class. In that case, remove one of the duplicates from the output LDIF schema file.

MODIFY
Modifies an existing attribute type or object class. For a **MODIFY** of an attribute type, the entire modified attribute type definition must be specified on one line. For a **MODIFY** of an object class, the entire modified object class definition must be specified. If the **MODIFY** specifies an attribute type or object class that does not exist in one of the schema input files, the **MODIFY** is ignored.

If any of the attribute types or object classes in the first column of any line denoting a **DELETE** or **MODIFY** are encountered in any of the input schema files, then the appropriate action is taken.

Notes:

1. Comments are denoted by a line starting with a # sign.
2. Blank lines are ignored.
3. When the **schema2ldif** utility encounters an erroneous line, the line is ignored and a warning messages is printed to standard out and the log file.

Following is an example of a **system.override** file:

```
#system.override file
top DELETE
faxnumber DELETE
telephone MODIFY attribute tphone ces telephone 32 critical
referral MODIFY objectclass refl
   requires
      objectClass
computer ADD attribute computer ces computer 50 normal
location_new ADD objectclass  location_new
   requires
      objectClass, ou
   allows
      businessCategory
```

-o *OIDFile*
Specifies an optional numeric object identifier file. The purpose of this file is to provide the numeric object identifiers for the attribute types and object classes which were specified in the input ***.at**, ***.oc**, or ***.conf** files.

The format of this file is shown here:

```
name                      2.5.4.41
country                   2.5.6.2
telephoneNumber           2.5.4.20
facsimileTelephoneNumber  2.5.4.23
```

The attribute type or object class name is the first field and the numeric object identifier is the second field.

Notes:

1. Comments in the OID file are denoted by a line starting with a # sign. Any line that contains more than two fields is an error. The utility will display an “ATTENTION” message on the console and ignore the erroneous line.
2. If an attribute type or object class name-numeric object identifier pair is specified twice in the OID file, the second numeric object identifier is used in the output LDIF file.

The z/OS LDAP server provides a default OID file called **name-noid.list**, which is located in the **/usr/lpp/ldap/etc** directory.

If you have additional attribute types or object classes in your schema files, be certain that any additional name-numeric object identifier pairs you add do not conflict with the object identifiers that are in the **name-noid.list** file.

-v Specifies verbose mode, where all debugging messages are printed to the log file. If the **-v** option is specified, the log file will contain messages that state what the **schema2ldif** utility has done to the input schema file or files. In this file, all of the attribute types and object classes that were formatted to LDIF format are denoted along with the input schema file name. Messages denoting that the optional input files (that is, any OID or override files) have been parsed are also written to the log file.

Examples

Following is an example attribute type file named **at.in**:

```
attribute name                    cis    name          50   normal
attribute acceleratorCapabilities cis    accelCap      11   normal
attribute accessHint              dn     accessHint  1000   normal
```

Figure 18. Example attribute type file (at.in)

Following is an example object class file named **oc.in**:

```
objectclass   cacheObject
        requires
                objectClass
        allows
                ttl

objectclass   cimBIOSelement
        requires
                objectClass
        allows
                primaryBIOS
```

Figure 19. Example object class file (oc.in)

The examples of **schema2ldif** that follow use these input files.

1. To convert **at.in** and **oc.in** to LDIF, use the following command:

   ```
   schema2ldif -s at.in oc.in -d output.ldif -l log.out
   ```

   This **schema2ldif** utility produces the following in the **output.ldif** file.

   ```
   dn: cn=schema,<namingcontext>
   changetype:modify
   add:x
   ```

```
attributetypes: ( name-at-oid NAME 'name' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  EQUALITY caseIgnoreMatch )
ibmAttributeTypes: ( name-at-oid ACCESS-CLASS normal )
attributeTypes: ( acceleratorCapabilities-at-oid NAME 'acceleratorCapabilities' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  EQUALITY caseIgnoreMatch )
ibmAttributeTypes: ( acceleratorCapabilities-at-oid ACCESS-CLASS normal )
attributeTypes: ( accessHint-at-oid NAME 'accessHint' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12  EQUALTIY distinguishedNameMatch )
ibmAttributeTypes: ( accessHint-at-oid ACCESS-CLASS normal )
objectclasses: ( cacheObject-oc-oid NAME 'cacheObject' SUP TOP MUST ( objectClass ) MAY ( ttl ) )
objectclasses: ( cimBIOSelement-oc-oid NAME 'cimBIOSelement' SUP TOP MUST ( objectClass ) MAY ( primaryBIOS ) )
```

Note that this generated LDIF file cannot be used directly with the z/OS LDAP server because the numeric object identifiers need to be added manually.

2. This example uses both **at.in** and **oc.in** as schema input files, but specifies an OID file named **oid.in** shown below.

```
name            2.5.4.41
accessHint      1.3.18.0.2.4.292
cacheObject     1.3.6.1.4.1.250.3.18
```

The following **schema2ldif** command:

```
schema2ldif -s at.in oc.in -d mixed.out -l log.out -o oid.in
```

produces the following output in **mixed.out**:

```
dn: cn=schema,<namingcontext>
changetype:modify
add:x
attributeTypes: ( 2.5.4.41 NAME 'name' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  EQUALITY caseIgnoreMatch )
ibmAttributeTypes: ( 2.5.4.41 ACCESS-CLASS normal )
attributeTypes: ( acceleratorCapabilities-at-oid NAME 'acceleratorCapabilities' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  EQUALITY caseIgnoreMatch )
ibmAttributeTypes: ( acceleratorCapabilities-at-oid ACCESS-CLASS normal )
attributeTypes: ( 1.3.18.0.2.4.292 NAME 'accessHint' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12  EQUALTIY distinguishedNameMatch)
ibmattributeTypes: ( 1.3.18.0.2.4.292 ACCESS-CLASS normal )
objectclasses: ( 1.3.6.1.4.1.250.3.18 NAME 'cacheObject' SUP TOP MUST ( objectClass ) MAY ( ttl ) )
objectclasses: ( cimBIOSelement-oc-oid NAME 'cimBIOSelement' SUP TOP MUST ( objectClass ) MAY ( primaryBIOS ) )
```

3. This example specifies a **slapd.conf** file (shown below) as input to the **schema2ldif** utility.

```
include  at.in
include  oc.in
sslKeyRingFile  key.kdb
sslKeyRingFilePW  none
sslCipherSpecs  12288
```

The following **schema2ldif** command:

```
schema2ldif -s slapd.conf -d mixed.out -l mixed.log -v
```

produces the following output in **mixed.out**:

```
dn: cn=schema,<namingcontext>
changetype:modifyadd:x
attributeTypes: ( name-at-oid NAME 'name' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  EQUALITY
caseIgnoreMatch )
ibmAttributeTypes: ( name-at-oid ACCESS-CLASS normal )
attributeTypes: ( acceleratorCapabilities-at-oid NAME 'acceleratorCapabilities' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  EQUALITY caseIgnoreMatch )
ibmAttributeTypes: ( acceleratorCapabilities-at-oid ACCESS-CLASS normal )
attributeTypes: ( accessHint-at-oid NAME 'accessHint' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12  EQUALITY distinguishedNameMatch )
ibmAttributeTypes: ( accessHint-at-oid ACCESS-CLASS normal )
objectclasses: ( cacheObject-oc-oid NAME 'cacheObject' SUP TOP MUST ( objectClass ) MAY ( ttl ) )
objectclasses: ( cimBIOSelement-oc-oid NAME 'cimBIOSelement' SUP TOP MUST ( objectClass ) MAY ( primaryBIOS ) )
```

The log file **mixed.log** will have the following information since the **-v** setting has been turned on:

```
PROCESSING input file slapd.conf
PROCESSING include file at.in
FORMATTING attribute type name
FORMATTING attribute type acceleratorCapabilities
FORMATTING attribute type accessHint
PROCESSING include file oc.in
FORMATTING objectclass cacheObject
FORMATTING objectclass cimBIOSelement
```

4. This example uses an override file. The **slapd.conf** from Example 3, the **at.in** file from Figure 18 on page 187, and the **oc.in** file from Figure 19 on page 187 files are used in this example. Following is the **system.override** file also used in this example:

```
# system.override file
name DELETE
cacheObject DELETE
owner ADD attribute owner dn owner 1000 normal
accessHint MODIFY attribute hint dn hint 1000 normal
cimBIOSelement MODIFY objectclass BIOSelement
   requires
      objectclass
   allows
      primaryBIOS
```

The following **schema2ldif** command:

```
schema2ldif -s slapd.conf -d mixed.out -l mixed.log -v -f system.override
```

produces the following output in **mixed.out**:

```
dn: cn=schema,<namingcontext>
changetype:modify
add:x
attributeTypes: ( owner-at-oid NAME 'owner' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 EQUALITY distinguishedNameMatch )
ibmAttributeTypes: ( owner-at-oid ACCESS-CLASS normal )
attributeTypes: ( acceleratorCapabilities-at-oid NAME 'acceleratorCapabilities'
  SYNTAX 1.3.6.1.1466.115.121.1.15 EQUALITY caseIgnoreMatch )
ibmAttributeTypes: ( acceleratorCapabilities-at-oid ACCESS-CLASS normal )
attributeTypes: ( hint-at-oid NAME 'hint' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 EQUALITY  distinguishedNameMatch )
ibmAttributeTypes: ( hint-at-oid ACCESS-CLASS normal )
objectclasses: ( BIOSelement-oc-oid NAME 'BIOSelement' SUP TOP MAY ( primaryBIOS ) )
```

The **mixed.log** file produces the following output:

```
ADDING: new attribute (owner) from system.override file to ADD table
ADDING: name from override file system.override to DELETE table
ADDING: cacheObject from override file system.override file to DELETE table
ADDING: accessHint from override file system.override to MODIFY table
ADDING: cimBIOSelement from override file system.override to MODIFY table
PROCESSING input file slapd.conf
PROCESSING include file at.in
SKIPPING attribute type name.  This attribute type was included as a DELETE in the override file.
FORMATTING attribute type owner
FORMATTING attribute type acceleratorCapabilities
USING modified attribute type hint from override file
FORMATTING hint
PROCESSING include file oc.in
SKIPPING objectclass type cacheObject. This objectclass was included as a DELETE in the override file.
USING modified object class BIOSelement from override file
FORMATTING BIOSelement
```

Note

Some changes to the resulting schema LDIF file may be required depending on the level of the schema files used during migration.

- Check for multiple definitions of the **cRLDistributionPoint**, **pilotDSA**, and **qualityLabelledData** object classes. Remove the duplicate definition from the LDIF file.
- Add **SUP person** to the definition of the **organizationalPerson** object class.

Chapter 15. Modify DN Operations

The Modify DN Operation allows a client to change the leftmost (least significant) component of the name of an entry in the directory, or to move a subtree of entries to a new location in the directory. This chapter explains the function of the Modify DN operation and the options supported to influence the scope and duration of the operation. In addition, it instructs on the techniques necessary to achieve certain forms of directory renames and movement, and it advises on issues which may result in unintentional or unwanted results.

| In z/OS LDAP, modify DN operations are only supported in the TDBM (DB2-based) backend.

Modify DN Operation Syntax

The z/OS implementation of the Modify DN operation supports all required and optional parameters described for the operation in RFC 2251. Specifically, these parameters are required:

- **entryDN:** This is the Distinguished Name (DN) of the entry whose name will be changed. This entry may or may not have subordinate entries. This parameter may not be a zero-length string.
- **newRdn:** The Relative Distinguished Name (RDN) that will form the leftmost component of the new name of the entry. This parameter may not be a zero-length string. If the intent of the Modify DN operation is to move the target entry to a new superior without changing its RDN, the old RDN value must be supplied in the **newRdn** parameter. The attributes and values in the **newRdn** parameter are added to the entry if they are not already present in the entry.
- **deleteoldrdn:** A boolean parameter that controls whether the old RDN attribute values are to be retained attributes of the entry or whether they will be deleted from the entry.

The following parameter to the Modify DN operation is optional:

- **newSuperior:** The Distinguished Name (DN) of the entry which will become the immediate superior of the renamed entry (identified by the **entryDN** parameter). If this parameter is present, it may consist of a zero-length string or a non-zero-length string. See “Modify DN operations related to suffix DNs” on page 201 for more information on the use of a zero-length string for this parameter. A zero-length string value for this parameter (″″) will signify that the new superior entry is the root DN.

This operation also supports optional values, or controls, to influence the behavior of the operation. Two controls are supported (see Appendix D, “Supported server controls,” on page 459):

- **IBMModifyDNTimelimitControl:** This control causes the Modify DN operation to be abandoned if its duration exceeds the time limit represented by the control value expressed in seconds. No changes are made if the operation is abandoned. If the server’s time limit is less than the time limit requested in the control for this operation, the server’s time limit will take precedence. See “Configuration file options” on page 56 for more information on the servers time limit. This control is honored even if it is set by the admin DN for the server. When this control is present, it will **not** be propagated to the replica servers. (See “Modify DN operations and replication” on page 208 for more information about replication of Modify DN operations.)
- **IBMModifyDNRealignDNAttributesControl:** This control causes the server to search for all attributes whose attribute type is based on a DN syntax (designated by OID 1.3.6.1.4.1.1466.115.121.1.12) and whose values match any of the old DN values being renamed as part of the Modify DN operation, and to modify the old DN values to reflect the corresponding renamed DN attribute values. This includes modifications to two other attribute types which have constructed DN-type attribute values (those whose attribute syntax is not distinguished name but which may be used to store DN values). They are **aclEntry** and ownership **entryOwner** attributes. Updates to constructed DN types will be limited to these two attributes defined by the LDAP Directory Server. No changes will be made to any user constructed types.

  This control is an all-or-none operation in which the server attempts to realign all appropriately-matched DN attribute values in the TDBM backend. Users cannot limit the scope of values which should be

realigned. If a failure arises during the realignment operation, it realigns none of the values, and the Modify DN operation fails. No changes are made if the operation is abandoned. It should be noted that even if the control is designated as non-critical, the server will still try to honor the intent of the control and if this attempt fails, the entire Modify DN operation will fail.

When **IBMModifyDNRealignDNAttributesControl** is present on a request to a master server on which replication of Modify DN operations is enabled, it will be propagated to the replica servers. (See “Modify DN operations and replication” on page 208 for more information about replication of Modify DN operations.)

A few simple examples of the use of the Modify DN operation follow. Each request will be expressed in the format of the ModifyDNRequest defined in RFC 2251, as well as in the corresponding invocation command for the z/OS client utility program **ldapmodrdn**. Refer to the *z/OS Integrated Security Services LDAP Client Programming* for more information on the **ldapmodrdn** utility.

Example 1: Simple Modify DN of leaf node

```
ModifyDNRequest ::= {
entry            cn=Kevin Heard, o=Athletics, o=Human Resources, o=Deltawing, c=AU
newrdn           cn=Kevin T. Heard
deleteoldrdn     TRUE
}
```

```
ldapmodrdn -h ldaphost -p ldapport -D binddn -w passwd -r "cn=Kevin Heard, o=Athletics, o=Human Resources,
  o=Deltawing, c=AU" "cn=Kevin T. Heard"
```

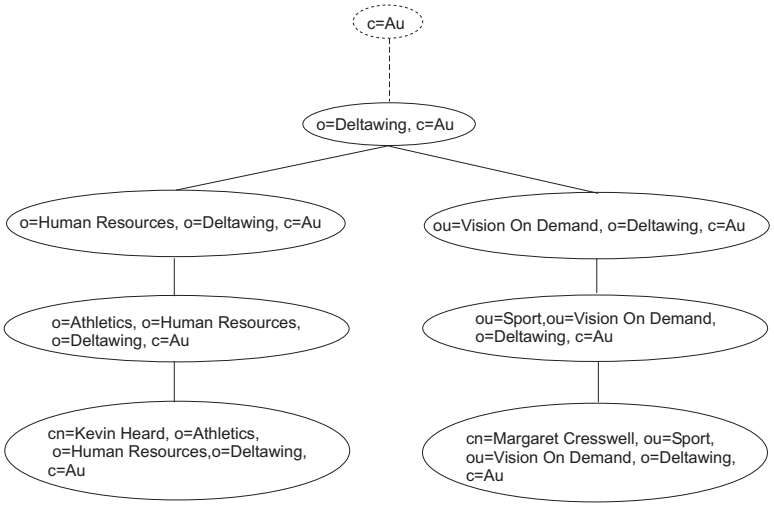


Figure 20. Before Modify DN operation

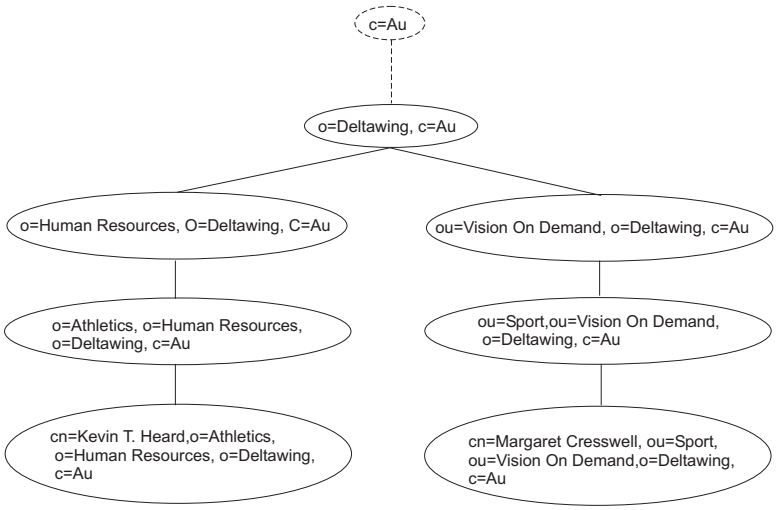


Figure 21. After Modify DN operation

Note: The **-r** parameter specifies that the old RDN attribute value (`cn=Kevin Heard`) will be deleted from the target entry after this operation.

Example 2: Simple Modify DN of non-leaf node

```
ModifyDNRequest ::= {
 entry             o=Athletics, o=Human Resources, o=Deltawing, c=AU
 newrdn            ou=College Athletics Dept.,
 deleteoldrdn    FALSE
 }


ldapmodrdn -h ldaphost -p ldapport -D binddn -w passwd "o=Athletics,
 o=Human Resources, o=Deltawing, c=AU" "ou=College Athletics Dept."
```

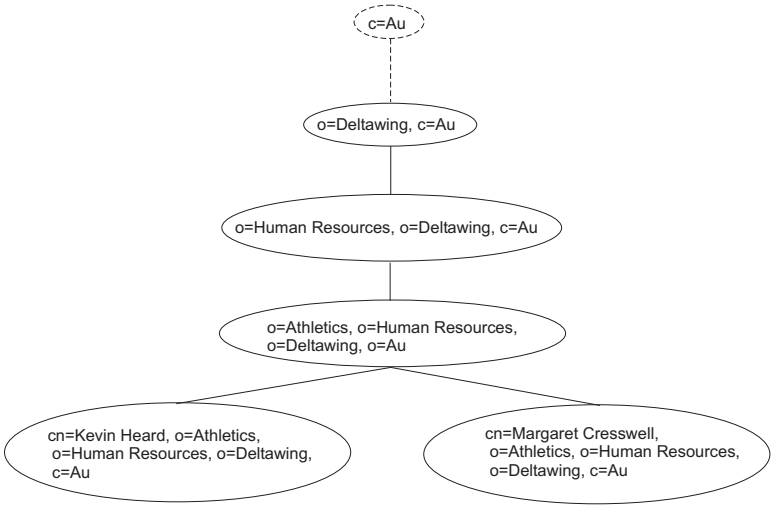


Figure 22. Before Modify DN operation

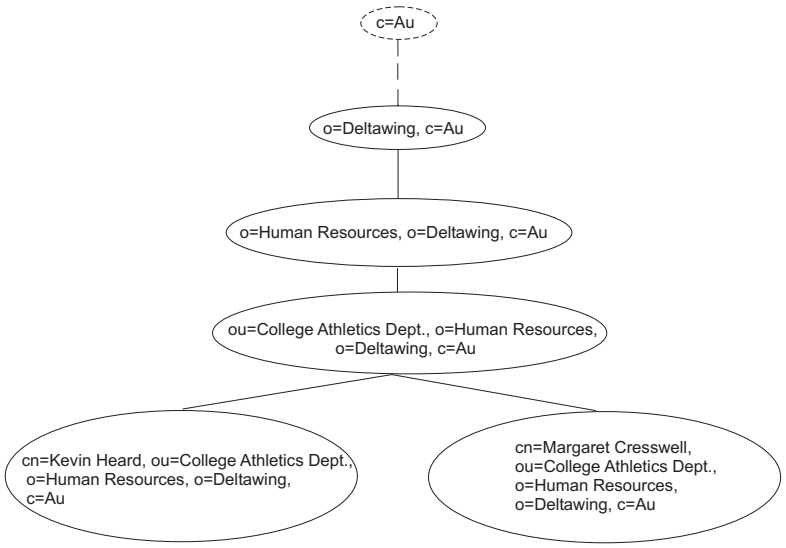


Figure 23. After Modify DN operation

Note: The absence of the **-r** parameter specifies that the old RDN attribute value (`o=Athletics`) will be preserved in the target entry after this operation.

Example 3: Modify DN of non-leaf node with relocation (newSuperior)

```
ModifyDNRequest ::= {
entry            o=Athletics, o=Human Resources, o=Deltawing, c=AU
newrdn           o=Adult Athletics
deleteoldrdn     FALSE,
newSuperior      ou=Sport, ou=Vision On Demand, o=Deltawing, o=AU
                 }
```

```
ldapmodrdn -h ldaphost -p ldapport -D binddn -w passwd -s "ou=Sport, ou=Vision On
 Demand, o=Deltawing, c=AU" "o=Athletics,o=Human Resources, o=Deltawing, c=AU" "o=Adult Athletics"
```

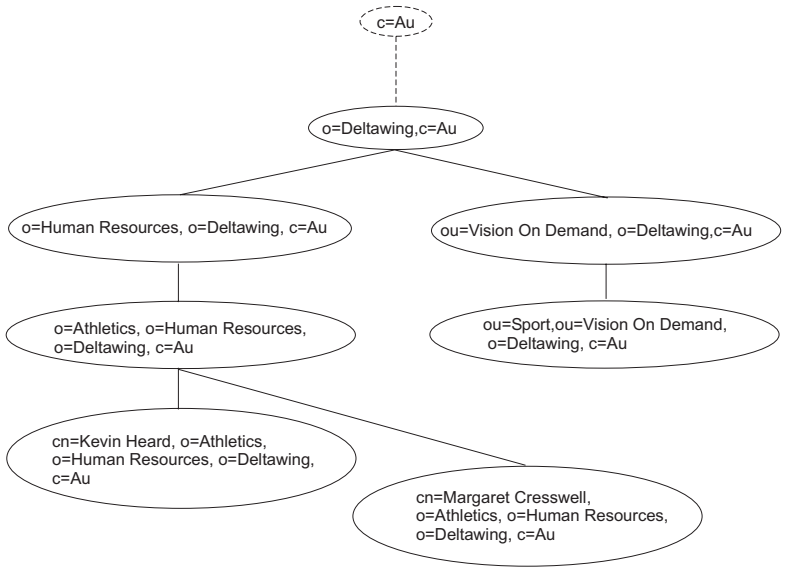


Figure 24. Before Modify DN operation

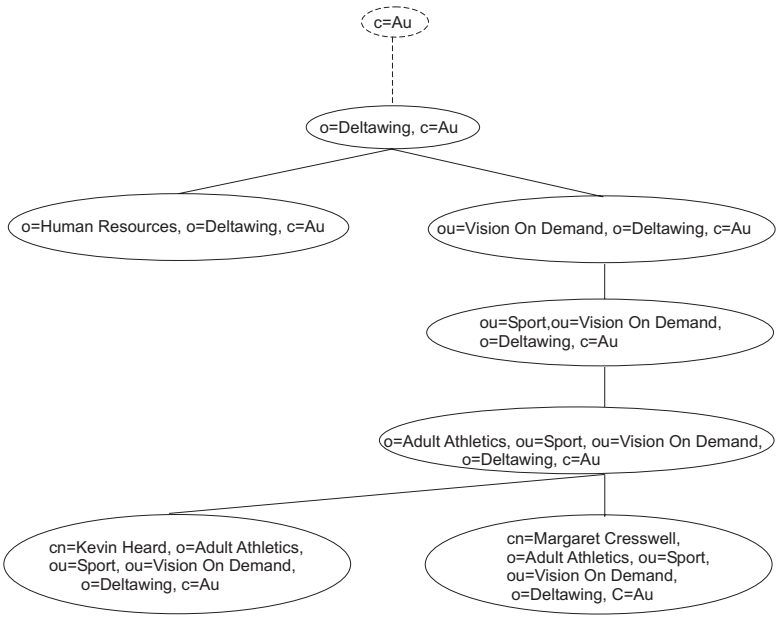


Figure 25. After Modify DN operation

Note: The absence of the **-r** parameter specifies that the old RDN attribute value (`o=Athletics`) will be preserved in the target entry after this operation. The target entry and descendants in its subtree will be relocated in the directory hierarchy.

Considerations in the use of Modify DN operations

As this operation has the potential to significantly change directory data and how it can be accessed, it is important that the user fully understand the data before using the Modify DN operation. Specifically, the user needs to know that:

- The ability of this operation to move directory subtrees has the potential for affecting many entries in the directory in a single operation.
- Certain options may result in modification of additional directory entries which are outside the scope of the directory subtrees being moved. This chapter will explain and give examples of how that can occur.
- Because the changes performed to the directory as a result of the operation are committed as a single transaction (or reversed if an error occurs), it may result in a long-running transaction, which may reduce concurrency of other LDAP operations targeted for the same directory entries. See “Concurrency considerations between Modify DN operations and other LDAP operations” on page 196 for more information.
- The scope of the changes may result in unanticipated effects in the directory and may affect user access to these entries. See “Access control changes” on page 199 for more information.
- There are limitations to which directory entries are eligible for the Modify DN operation. See “Eligibility of entries for rename” on page 196 for more information.
- In case the directory needs to be returned to a state prior to a Modify DN operation, the directory should be backed up by using the **tdbm2ldif** utility program or by using DB2 utilities to generate a DB2 image copy of the underlying tablespaces. See Chapter 11, “Running and using the LDAP backend utilities,” on page 135 for more information about the **tdbm2ldif** utility program, and *DB2 Utility Guide and Reference* for more information about the DB2 image copy. In addition to backing up the directory contents, activity logging should be enabled before nontrivial changes are made to the directory. See “Activity log and console listing of Modify DN operations” on page 211 for more information.
- There are considerations if the data to be modified by this operation is being replicated. See “Modify DN operations and replication” on page 208 for more information.

Eligibility of entries for rename

Entries in the directory which are targeted to be renamed in a single Modify DN operation are subject to these constraints:

1. All entries to be renamed must be located in the same TDBM backend instance targeted by the Modify DN operation. The Modify DN operation with *newSuperior* option will move subtree entries within the same TDBM backend instance, and will not permit movement of subtree entries from one backend instance to another. The operation will only relocate entries to subtrees conforming to the same schema. The entry to be renamed must exist in the backend, and the new DN for the entry must not already exist in the backend.
2. Referral entries may be renamed as part of a Modify DN operation. If a referral entry is renamed as part of a Modify DN operation, its corresponding entry in another backend must be manually updated to reflect the name changes; no automatic updates are propagated to those backends from the target backend. Referrals which exist in other directory servers which refer to any of the entries whose DNs were modified in the local directory by a Modify DN operation will need to be manually updated to reflect the changes; no automatic updates are propagated to those servers from the local one.
3. Schema entries may not be renamed explicitly by the Modify DN operation. However, if a TDBM suffix entry is renamed, and the schema entry DN included that suffix, then the schema entry will be automatically renamed to reflect the new DN.
4. Entries renamed by a Modify DN operation must conform to the schema rules for the backend in which the operation occurs. As such, the RDN attribute type must be consistent with the schema rules for the object classes of the entry: a Modify DN operation will fail if the attribute type of *newRdn* is not in the **MUST** or **MAY** list for the entry's object classes.
5. When **IBMModifyDNRealignDNAttributesControl** is present on a Modify DN request, the operation modifies occurrences of the renamed DN if the syntax for the attribute containing the matching DN value is distinguished name. If a subtree of entries is moved and the entries are thereby renamed, then the operation modifies all occurrences of all renamed DNs in the. In addition, if a matching DN value has an attribute syntax which is one of the two constructed attribute types, **aclEntry** or **entryOwner**, the matching values will be replaced with their renamed values. Any user constructed attribute types which contain DNs whose values match those being modified by the Modify DN operation will be left unchanged.
6. A Modify DN operation can succeed even if the *newRdn* value already exists as an attribute value in the entry, as long as that value is not already the sole RDN attribute value. For example, suppose an entry with DN of `dept=AAA,ou=mydivision,o=MyCompany,c=us` is to be renamed with the *newRdn* `sector=northeast`, and that the entry already contains the multi-valued attribute **sector** which currently contains attribute values of `northeast` and `northcentral`. This rename will succeed, and the value `northeast` will appear only once for the attribute type **sector**.
7. Entries may be renamed only if all access control requirements are satisfied for the bound user, as determined by the effective ACL and ownership permissions for those entries and attributes. See "Access control and ownership" on page 197 for detailed explanation and examples of this effect.
8. Alias entries (entries containing the **aliasedObjectName** attribute and either the **alias** or **aliasObject** object class) can be renamed as part of a modify DN operation as long as this does not result in an **aliasedObjectName** value that is a DN equal to or below the DN of the renamed alias entry.

Concurrency considerations between Modify DN operations and other LDAP operations

The ability of the Modify DN operation to rename non-leaf nodes in the directory (which causes all entries which are hierarchical subordinates of the target entry to be renamed) and the ability to move directory subtrees have the potential for affecting many entries in the directory in a single operation. Use of **IBMModifyDNRealignDNAttributesControl** with this operation may further result in modification of additional directory entries which are outside the scope of the directory subtrees being renamed or moved.

Changes to all entries affected by the operation are committed at the same time. While modified entries are awaiting the transaction commit point, database locks are held which prevent other concurrent operations from sharing and modifying the data. If many entries undergo modification with this operation, it may result in a long-running transaction which has potential for reducing concurrency of other operations targeted for the same directory entries.

Although the LDAP server is capable of processing concurrent LDAP operations targeted at a given TDBM backend instance while the Modify DN operation is in progress, the extent to which such concurrency is possible will depend on what data in the directory may be needed and locked by the competing operations. In addition, if the number of entries being affected by the Modify DN operation is large or if the database is small, the underlying DB2 locking mechanism may escalate locking levels, which would result in more entries being excluded from use by other concurrent operations than just those which are modified by the Modify DN operation. It may be advisable to submit such a request during a low-activity period when demand for the same resources by multiple concurrent operations is relatively low.

For example, the Modify DN operation that is shown in 194 and 195would potentially be susceptible to lock contention when:

- there are concurrent update operations under the new parent ″ou=Sport, ou=Vision On Demand, o=Deltawing, c=AU″
- or there are concurrent update operations under the old parent ″o=Human Resources, o=Deltawing, c=AU″
- or DB2 locking chooses an access path that results in lock escalation for entries under ″o=Deltawing, c=AU″

For more information on DB2 lock escalation, see IBM redbook, *Locking in DB2 for MVS/ESA Environment (SG24–4725)*, at http://www.redbooks.ibm.com/ or *DB2 Application Programming and SQL Guide*.

Access control and ownership

For all entries being renamed, the caller must have **w**(rite) permissions for the attribute values that will have to change in all affected entries. In addition, if the *newSuperior* parameter is present on the Modify DN request, the caller must have permissions of **object:a** on the *newSuperior* entry and **object:d** on the target entry at the top of the subtree of entries being moved. If the caller lacks one or more of these permissions, the operation is denied. No access control checking is done against any of the target entry's subordinates even though their DN is changed. It should be noted that if the caller is an effective owner of any of the entries being renamed, the permissions are automatically satisfied for those entries.

In addition, if the **IBMModifyDNRealignDNAttributesControl** accompanies a Modify DN request, then the bound DN must have **w**(rite) permission to all of the attributes that are changed as a result of realignment of the DN values.

Example:

Assume our sample directory contains the following entry which will be the target of a Modify DN operation, and which contains explicit ACL information:

```
dn: o=Athletics, o=Human Resources, ou=Delta Home Media Ltd., o=Deltawing, c=AU
aclEntry: access-id: cn=Mark Crawford, ou=Human Resources, ou=Delta Home Media Ltd.,
 o=Deltawing,c=AU:normal:rswc:sensitive:rsc:object:d

(other attributes not shown)
```

The directory also contains an entry with DN `ou=Production, ou=Vision On Demand,o=Deltawing, c=AU` which will be the newSuperior of the Modify DN operation. This entry inherits the following ACL information (propagated from a superior entry):

```
aclEntry: access-id: dn: cn=Mark Crawford, ou=Human Resources, ou=Delta Home Media
 Ltd., o=Deltawing, c=AU:normal:rwsc:sensitive:rsc:object:a
```

In addition, there are several entries containing attributes of DN syntax. For this example, assume that these attribute types and their respective attribute access classes are as follows:

attribute: reportingOrganization
access-class: sensitive

attribute: workingOrganization
access-class: normal

The LDIF format representation of the entries containing **reportingOrganization** or **workingOrganization** attributes are:

```
dn: cn=Lisa Fare, ou=Human Resources, ou=Delta Home Media Ltd., o=Deltawing, c=AU
cn: Lisa Fare
objectclass: organizationalPerson
objectclass: person
objectclass: TOP
aclEntry: access-id: cn=Mark Crawford, ou=Human Resources, ou=Delta Home
Media Ltd., o=Deltawing,c=AU:normal:rsc:sensitive:rs
sn: Fare
title: Occupational Health and Safety Administrator
telephonenumber: (07) 635 1432
manager: cn=John Gardner, ou=Human Resources Group, ou=Deltawing InfoSystems,
 o=Deltawing, c=AU
secretary: cn=Ian Campbell, o=Deltawing, c=AU
reportingOrganization: o=Athletics, o=Human Resources, ou=Delta Home Media Ltd.,
 o=Deltawing, c=AU

dn: cn=Laurie Wood, ou=Human Resources Group, ou=Deltawing Automotive Ltd., o=Deltawing, c=AU
cn: Laurie Wood
objectclass: organizationalPerson
objectclass: person
objectclass: TOP
aclEntry: access-id: cn=Mark Crawford, ou=Human Resources, ou=Delta Home
 Media Ltd., o=Deltawing,c=AU:normal:rswc:sensitive:rsw
sn: Wood
telephonenumber: (03) 9335 2114
title: Pay Officer
workingOrganization: o=Athletics, o=Human Resources, ou=Delta Home Media Ltd.,
 o=Deltawing, c=AU
```

Relocating an entry

User "cn=Mark Crawford, ou=Human Resources, ou=Delta Home Media Ltd., o=Deltawing,c=AU" submits the following Modify DN operation request to the server to relocate the target entry:

```
ldapmodrdn -h ldaphost -p ldapport -D "cn=Mark Crawford, ou=Human Resources, ou=Delta Home
 Media Ltd., o=Deltawing,c=AU" -w passwd -s "ou=Production, ou=Vision On Demand,
 o=Deltawing, c=AU" "o=Athletics, o=Human Resources, ou=Delta Home Media Ltd., o=Deltawing,
 c=AU" "o=Athletics Division"
```

The **-s** parameter specifying *newSuperior* is present on this operation request, so in addition to the access permissions needed for all Modify DN operations (**w** on affected attributes), the user also needs **object:d** on the target entry and **object:a** on the newSuperior entry. The bound user is in the **aclEntry** for the target entry as well as in the **aclEntry** for the newSuperior entry, and has all required access permissions (can write attributes and delete the target entry, and can add objects under the newSuperior entry), so the operation is permitted.

Relocating an entry with DN realignment requested

Now suppose the same user submits a Modify DN operation request to the server to relocate the same target entry under the same newSuperior entry, but with the addition of the control requesting realignment of DN attribute values (**-a** parameter):

```
ldapmodrdn -h ldaphost -p ldappart -D "cn=Mark Crawford, ou=Human Resources, ou=Delta Home
 Media Ltd., o=Deltawing,c=AU" -w passwd -a -s "ou=Production, ou=Vision On Demand, o=Deltawing, c=AU"
 "o=Athletics, o=Human Resources, ou=Delta Home Media Ltd., o=Deltawing, c=AU" "o=Athletics Division"
```

In addition to the permissions required on the previous example, this operation now requires additional permissions to be checked on entries containing values which qualify for realignment. The DN being modified (`"o=Athletics, o=Human Resources, ou=Delta Home Media Ltd., o=Deltawing, c=AU"`) is found in DN-syntax attributes of two entries: The entry with DN `"cn=Laurie Wood, ou=Human Resources Group, ou=Deltawing Automotive Ltd., o=Deltawing, c=AU"` contains this value in the **workingOrganization** attribute, and the entry with DN `"cn=Lisa Fare, ou=Human Resources, ou=Delta Home Media Ltd., o=Deltawing, c=AU"` contains this value in the **reportingOrganization** attribute.

The bound user is in the aclEntry for `"cn=Laurie Wood, ou=Human Resources Group, ou=Deltawing Automotive Ltd., o=Deltawing, c=AU"`. The **workingOrganization** attribute is in the access-class of normal, and the bound user is granted **w** access to this class of attributes, so the realignment of the DN value would be permitted in this entry.

The bound user is also in the **aclEntry** for `"cn=Lisa Fare, ou=Human Resources, ou=Delta Home Media Ltd., o=Deltawing, c=AU"`. The **reportingOrganization** attribute is in the access-class of `sensitive`, and the bound user is granted only **rs** permissions on **sensitive** attributes in the entry, so the realignment of this value would be denied. Even though the bound user had adequate permissions to perform the relocation of the target entry and had adequate permissions to perform realignment of the DN value in one of the two entries containing a matching DN, the operation would fail because the bound user does not have the necessary permissions on everything needed to complete the operation.

Access control changes

If a Modify DN operation is accompanied by the *newSuperior* parameter, changes in effective ACLs and in effective ownership of the relocated entries may result. Regardless of the effective ACLs which applied to the moved subtree in its old location, the moved subtree now inherits any propagating ACLs applying to the *newSuperior* entry. As a consequence, entries to which a user had access before the request may no longer be accessible by that user, and entries to which access was denied for a given user before the request may now be accessible by that user.

Explicit ACLs in the entry or subtree override propagating ACLs. All explicit ACLs which were in the moved subtree at its original location move along with the entries.

When renaming a DN, it is possible that ACLs and entryOwners containing the renamed DN will be modified. Therefore, prior to such a move or rename users should carefully consider how ownership and accessibility to entries protected by these attributes may change after the move, and what ACL and ownership changes may be desired, if any.

The following is an example of how a Modify DN operation might affect access controls:

```
ModifyDNRequest ::= {
entry            o=Athletics, o=Human Resources, o=Deltawing, c=AU
newrdn           o=Adult Athletics
deleteoldrdn     FALSE,
newSuperior      ou=Sport, ou=Vision On Demand, o=Deltawing,c=AU
                 }
```

```
ldapmodrdn -h ldaphost -p ldapport -D binddn -w passwd -s "ou=Sport,
 ou=Vision On Demand, o=Deltawing, c=AU" "o=Athletics,o=Human Resources,
 o=Deltawing, c=AU" "o=Adult Athletics"
```

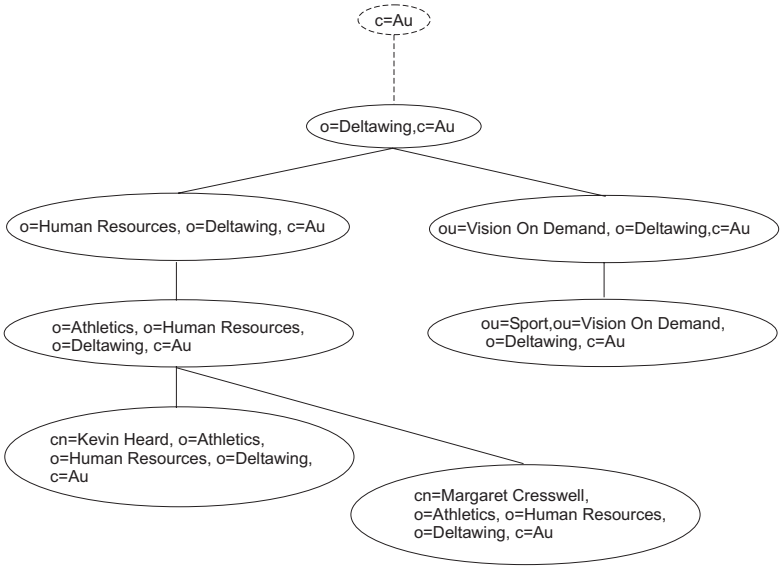


Figure 26. Before Modify Dn operation

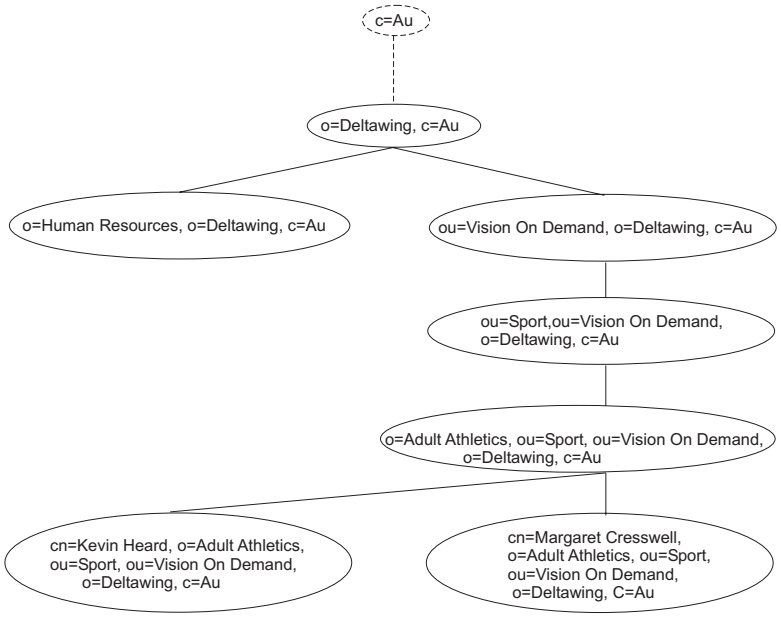


Figure 27. After Modify DN operation

Assume that the entry with DN `o=Human Resources, o=Deltawing, c=AU` has an explicit propagating ACL containing the following **aclEntry**:

```
aclEntry: access-id: cn=Mark Edmondson, ou=Vision On Demand, ou=Delta Home Media Ltd.,
o=Deltawing,c=au:normal:rwcs:sensitive:rwcs:critical:rws:object:d
```

Also, assume that the entry with DN `ou=Sport, ou=Vision On Demand, o=Deltawing, c=AU` has an explicit propagating ACL containing the following **aclEntry**:

```
aclEntry: access-id: cn=Mark Edmondson, ou=Vision On Demand, ou=Delta Home Media Ltd.,
 o=Deltawing,c=au:normal:rws:sensitive:r:critical:r:object:a
```

If the user bound as DN `cn=Mark Edmondson, ou=Vision On Demand, ou=Delta Home Media Ltd., o=Deltawing,c=AU` performs the example Modify DN operation, there are at least two consequences which should be noted:

- While this DN previously had **rwcs** permissions on sensitive attributes in the entry `o=Athletics, o=Human Resources, o=Deltawing, c=AU` and **rws** permissions on critical attributes in the same entry, this DN now has only **r** access on both sensitive and critical attributes in the entry after the relocation. It might be expected that a given DN will have the same accessibility to specific entries and data in the directory after a Modify DN operation as it had to those entries and data before the operation, but this example demonstrates that such an expectation is not valid.
- If, after completion of the Modify DN operation, the bound user decides that they wish to return the moved entry (and its subordinates) back to their original location in the directory hierarchy, this will not be possible with the access controls currently in place. The bound DN has only **object:d** permission on the old superior node (`"o=Human Resources, o=Deltawing, c=AU"`) where **object:a** is needed to effect the move of an entry or subtree under the superior node, and the bound DN has only **object:a** permission on the moved entry (`"o=Adult Athletics, ou=Sport, ou=Vision On Demand, O=Deltawing, c=AU"`) where **object:d** is needed to move the entry. Thus, while it may be expected that a given DN can reverse a Modify DN operation under all circumstances, this example demonstrates that such an expectation is not valid.

Ownership changes

When the *newSuperior* parameter accompanies the Modify DN request, any entries in a relocated subtree which had explicit owners prior to the relocation will preserve that explicit ownership after the relocation has been performed. Any entries in the relocated subtree which inherited ownership prior to relocation will continue to inherit ownership following relocation. If the owning entry prior to relocation was a node superior to the relocated entry, the owning entry will be the new superior entry. If the owning entry was an entry within the relocated subtree, the owning entry is preserved following the relocation.

Any entries in the relocated subtree which propagated ownership to subordinates prior to relocation continue to propagate ownership to subordinates after the relocation.

Refer to the example in the preceding section “Access control changes” on page 199.

Assume that the entry with DN `o=Human Resources, o=Deltawing, c=AU` has an explicit propagating owner of `cn=Mark Crawford, ou=Human Resources, ou=Delta Home Media Ltd.,o=Deltawing,c=AU`.

Also, assume that the entry with DN `ou=Sport, ou=Vision On Demand, o=Deltawing, c=AU` has an explicit propagating owner of `cn=Neville McAuliffe, ou=Human Resources Group, ou=Deltawing Infosystems, o=Deltawing, c=AU`.

Before the Modify DN operation, the effective owner of the renamed entry is `cn=Mark Crawford, ou=Human Resources, ou=Delta Home Media Ltd., o=Deltawing,c=AU`; after completion of the operation, the effective owner of the renamed entry is now `cn=Neville McAuliffe, ou=Human Resources Group, ou=Deltawing Infosystems, o=Deltawing,c=AU`. Thus, the act of relocating an entry may change the effective owner of that entry and of its subordinates.

Modify DN operations related to suffix DNs

The Modify DN operation may be used to modify the DNs of any and all entries in a given TDBM backend. In addition to renaming leaf entries (directory entries with no subordinate entries) and mid-hierarchy entries (directory entries which have both superior entries and subordinate entries), suffix entries may also be renamed. Suffix entries may be renamed to become non-suffix entries and suffix entries may be renamed such that they continue to be suffix entries. In addition, non-suffix entries may be renamed to become

suffix entries. This section provides example scenarios for rename operations which involve suffix entries. It summarizes constraints which have been adopted for the z/OS LDAP implementation which are not defined in the protocol behavior prescribed by RFC 2251 for the Modify DN operation. Examples are provided on how various renaming scenarios may be accomplished, and factors to be considered when performing these operations are discussed.

Scenario constraints

When using a Modify DN operation to move or rename a suffix DN, numerous scenarios should be considered. A Modify DN operation must be completely contained within a given instance of a TDBM backend. Both source entries and target entries must be in the same backend. Each of the sample renaming scenarios which follow requires that the entry must exist and the result of a rename must not yield a new entry which already exists in this backend instance. If either one or both of these assumptions are violated, the operation fails.

Several constraints will apply which are not defined by RFC 2251 in the description of the protocol behavior:

1. If an entry being renamed will become (or remain) a suffix, the new DN must be designated in the server's configuration file as a suffix for the backend, otherwise the operation will not be permitted.
2. The *newRdn* parameter of the Modify DN request must contain a non-null value, otherwise the operation request will be treated as an error.
3. If the *newSuperior* parameter is present, it may contain a zero-length string signifying that the *newSuperior* entry is the root DN.

In the directory hierarchy diagrams which follow, a circle outlined with a dashed line represents a component of a suffix DN. Circles containing gray fill represent DNs for which an entry exists in the directory.

Example scenarios

The following are example scenarios:

1. Rename a suffix RDN with no accompanying newSuperior, and the new DN remains a suffix after the rename is completed.

   Example:

   Suffixes defined in the server configuration file:
   suffix: `ou=End_GPL, o=MyCompany, c=US`
   suffix: `ou=Endicott, o=MyCompany, c=US`

   Rename operation is to rename suffix object entry `ou=End_GPL, o=MyCompany, c=US` to suffix object entry `ou=Endicott, o=MyCompany, c=US`

   The following figure is an example of this operation:

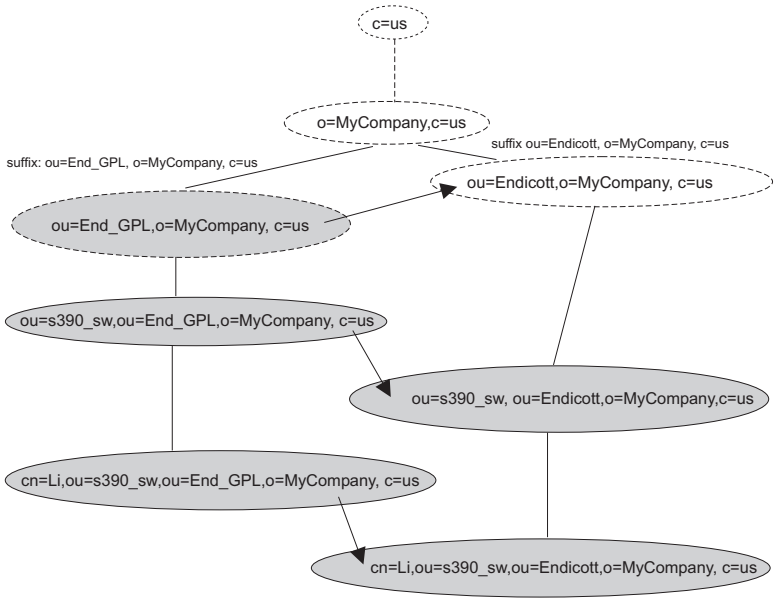


Figure 28. Suffix rename with no new superior

The new DN must be already designated as a suffix for this backend, otherwise this operation will fail.

The operation is performed the same as a rename of any other RDN in the directory

a. Send Modify DN operation request with
   target=`ou=End_GPL, o=MyCompany, c=US`
   newRdn=`ou=Endicott`

This results in renaming `ou=End_GPL, o=MyCompany, c=US` to `ou=Endicott, o=MyCompany, c=US` and in renaming subordinate entries accordingly.

2. Rename of suffix DN with an accompanying newSuperior, and the new DN remains a suffix after the rename is completed. Example:

   Suffix defined in the server configuration file:
   suffix: `ou=Endicott, o=MyCompany, c=us`

   Rename operation is to rename suffix object entry `ou=Endicott, o=MyCompany, c=us` to suffix object entry `o=MyCompany, c=us`

   The following figure shows an example of this operation:

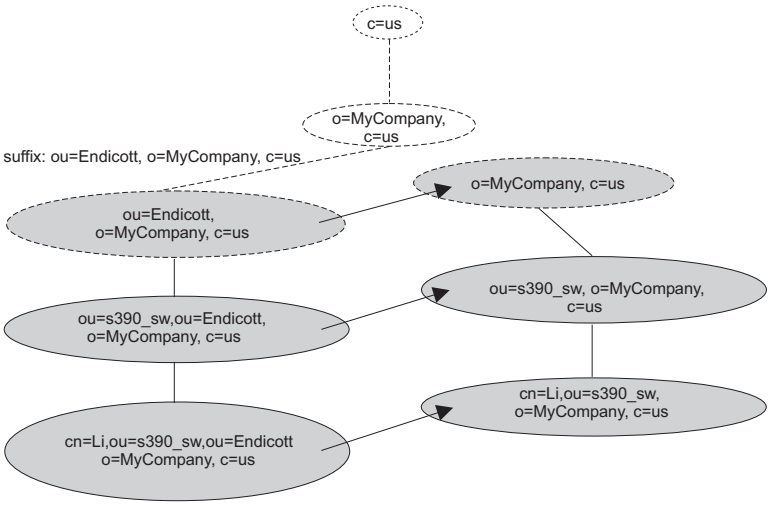


Figure 29. Suffix rename with new superior

This scenario, which involves renaming an existing suffix to an overlapping new suffix, must be performed in several steps, since the product does not permit designation in the server configuration file of overlapping suffixes for the same backend instance. The definition of overlapping suffixes is when two suffixes with differing numbers of naming components are equal to the extent of the shorter of the two suffixes. For example, `ou=Endicott, o=MyCompany, c=US` and `o=MyCompany,c=US` are considered to be overlapping suffixes, while `ou=Endicott, o=MyCompany, c=US` and `ou=Raleigh, o=MyCompany, c=US` are not considered to be overlapping suffixes.

This rename can be accomplished by having a temporary suffix pre-defined for the backend (for example, `o=OurTemporarySuffix`), renaming the target entry to become the temporary suffix, stopping the server and deleting the suffix `ou=Endicott, o=MyCompany, c=us` and adding the suffix `o=MyCompany, c=us`, and restarting the server. The temporary suffix would later be deleted from the list of suffixes for the backend.

a. Send a Modify DN operation request with
   target= `ou=Endicott, o=MyCompany, c=us`
   newRdn= `o=OurTemporarySuffix`
   newSuperior= `"" (present in request with zero-length string)`

   This results in renaming `ou=Endicott, o=MyCompany, c=us` to `o=OurTemporarySuffix`. Note that the server treats *newRdn* as an error if it contains a zero-length string, but zero-length strings are permitted in the *newSuperior* argument to signify that the superior entry is the root DN.

b. Stop server, remove suffix `ou=Endicott, o=MyCompany, c=us` from the server configuration file, add suffix `o=MyCompany, c=us`, and restart server.

   This results in adding the desired target suffix without a resulting conflict from overlapping suffixes.

c. Send a Modify DN operation request with
   target= `o=OurTemporarySuffix`
   newRdn= `o=MyCompany`
   newSuperior= `c=us`

   This step results in renaming the temporary suffix `o=OurTemporarySuffix` to the desired suffix `o=MyCompany, c=us`, thereby accomplishing the rename from `ou=Endicott, o=MyCompany, c=us` to `o=MyCompany, c=us`. In the process, subordinate entries would be renamed accordingly.

3. This example shows the renaming of a suffix to another overlapping suffix higher in the directory hierarchy. A similar scenario could also be performed involving the rename of a suffix to another overlapping suffix, where the new name is a suffix lower in the directory hierarchy. Example:

   Suffix defined in the server configuration file
   suffix: `ou=Endicott, o=MyCompany, c=us`

Rename operation is to rename suffix entry `ou=Endicott, o=MyCompany, c=us` to suffix object entry `div=S390, ou=Endicott, o=MyCompany, c=us`

The following figure shows an example of this operation:
This rename can be accomplished by having a temporary suffix pre-defined for this backend in the

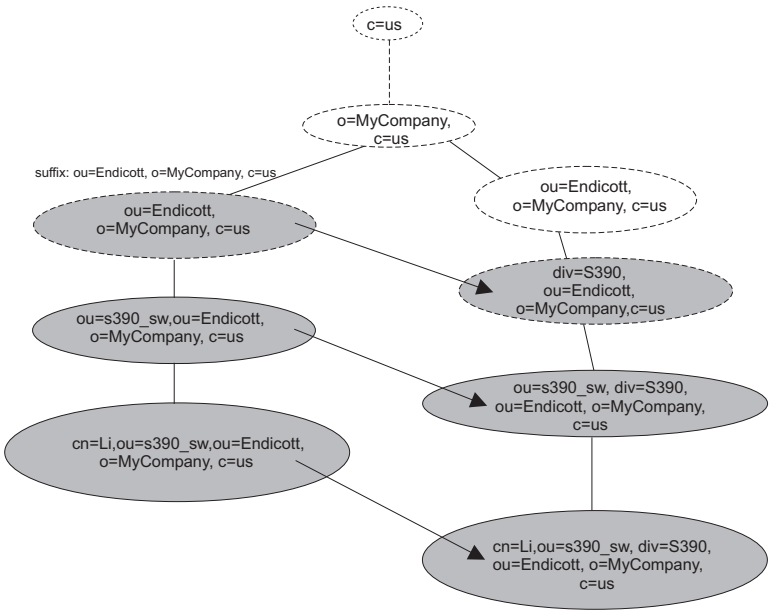


Figure 30. Overlapping suffix rename A

server configuration file (for example, `o=OurTemporarySuffix`), renaming the target entry to become the temporary suffix, stopping the server and deleting the suffix `ou=Endicott, o=MyCompany, c=us` and adding the suffix `div=S390, ou=Endicott, o=MyCompany, c=us`, and restarting the server. The temporary suffix would later be deleted from the list of suffixes for the backend. This scenario would be done like so:

a. Send a Modify DN operation request with
   target= `ou=Endicott, o=MyCompany, c=us`
   newRdn= `o=OurTemporarySuffix`
   newSuperior= `"" (present in request with zero-length string)`
b. Stop server, remove suffix `ou=Endicott, o=MyCompany, c=us`, add suffix `div=S390, ou=Endicott, o=MyCompany, c=us`, and restart server.
c. Send a Modify DN operation request with
   target= `o=OurTemporarySuffix`
   newRdn= `div=S390`
   newSuperior= `ou=Endicott, o=MyCompany, c=us`

At this point, it should be noted that if these operational scenarios are to be replicated from a master server to one or more replica servers, there is a procedure which must be followed to permit this.

a. Stop the replica server(s), add the temporary suffix (`o=OurTemporarySuffix` in our examples), restart the replica server(s).
b. Stop the master server, perform the previous Steps 3a and 3b from the examples above. This will result in the intermediate rename to be performed on the master server and the results to be propagated to the replica server(s).
c. Stop the replica server(s), delete the original suffix (`ou=Endicott, o=MyCompany, c=us` in both examples above), add the new suffix (`o=MyCompany, c=us` in the first example above, `div=S390, ou=Endicott, o=MyCompany, c=us` in the second example above), and restart the replica server(s).

d. Perform the previous Step 3c from the examples above. This will result in the rename of entries to the final destination on the master server and in the results being propagated to the replica server(s).

4. Rename of suffix DN (some component other than RDN), and the new DN remains a suffix after the rename is completed. Example:

   Suffixes defined in the server configuration file:
   suffix: `ou=Endicott, o=MyCompany, c=us`
   suffix: `ou=Endicott, o=MyCompany_ny, c=us`

   Rename operation is to rename suffix entry object `ou=Endicott, o=MyCompany, c=us` to suffix entry object `ou=Endicott, o=MyCompany_ny, c=us`

   The following figure shows an example of this operation:
   The new DN must be already designated as a suffix for this backend, otherwise this operation will fail.

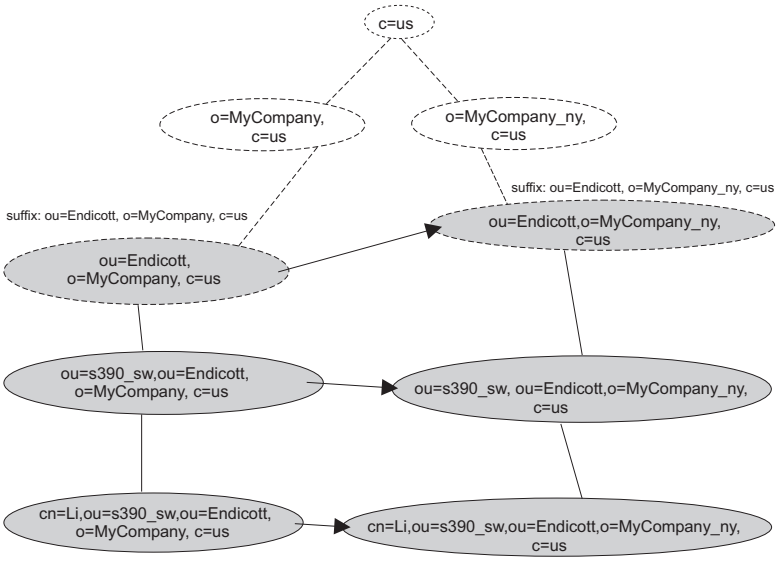


Figure 31. Overlapping suffix rename B

   The operation is performed the same as a rename of any other DN in the directory. The product will permit the rename to occur in one step, even if an object entry for newSuperior does not already exist, since the newly-named object will become a suffix entry.

   a. Send a Modify DN operation request with
      target= `ou=Endicott, o=MyCompany, c=us`
      newRdn= `ou=Endicott`
      newSuperior= `o=MyCompany_ny, c=us`

      This results in renaming the DN from `ou=Endicott, o=MyCompany, c=us` to `u=Endicott, o=MyCompany_ny, c=us` and in renaming subordinate entries accordingly.

5. Rename of suffix DN (including some component other than RDN), with an accompanying newSuperior, but the new DN is no longer a suffix Example:

   Suffixes defined in the server configuration file:
   suffix: `ou=End, o=MyCompany, c=us`
   suffix: `ou=End,ou=MyCompany_na, o=MyCompany, c=us`

   Rename operation is to rename suffix entry object `ou=End, o=MyCompany, c=us` to non-suffix entry object `ou=GPL, ou=End, ou=MyCompany_na, o=MyCompany, c=us`

The following figure shows and example of this operation:
The newSuperior entry object must already exist before this operation will be permitted.

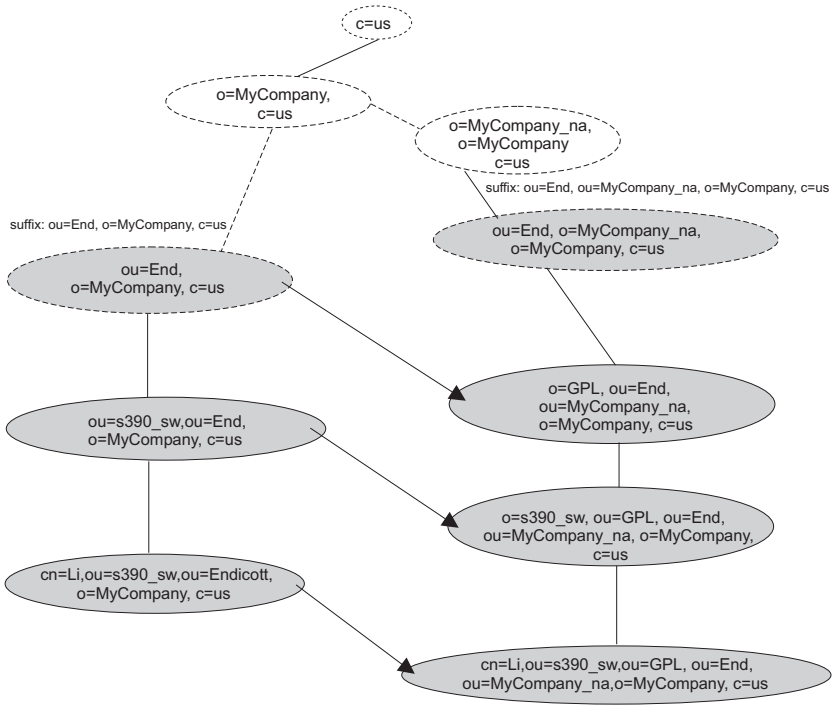


Figure 32. Suffix rename to non-suffix entry

a. Send a Modify DN operation request with
   target= `ou=End, o=MyCompany, c=us`
   newRdn= `ou=GPL`
   newSuperior= `ou=End, ou=MyCompany_na, o=MyCompany,c=us`

   This results in renaming `ou=End, o=MyCompany, c=us` to `ou=GPL, ou=End, ou=MyCompany_na, o=MyCompany, c=us` and in renaming subordinate entries accordingly.

6. Rename of a non-suffix DN (including some component other than RDN), with an accompanying newSuperior, and the new DN is now a suffix Example:

   Suffixes defined in the server configuration file:
   suffix: `ou=End, o=MyCompany, c=us`
   suffix: `o=Lotus, c=us`

   Rename operation is to rename non-suffix `div=Lotus,ou=End, o=MyCompany, c=us` to suffix `o=Lotus, c=us`

   The following figure shows and example of this operation:

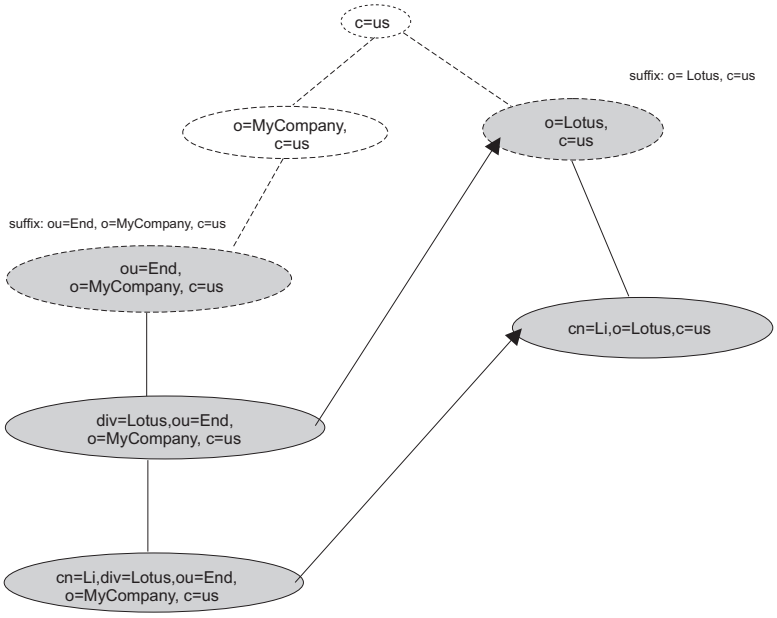


Figure 33. Rename non-suffix entry to suffix entry

The new DN must be already designated as a suffix for this backend, otherwise this operation will fail.

a. Send a Modify DN operation request with
   target= `div=Lotus,ou=Endicott, o=MyCompany, c=us`
   newRdn= `o=Lotus`
   newSuperior= `c=us`

   This step results in renaming `div=Lotus,ou=Endicott, o=MyCompany, c=us` to `o=Lotus, c=us` and in renaming subordinate entries accordingly.

Modify DN operations and replication

Modify DN operations may be classified into two categories:

1. Simple Modify DN operations are those which rename a leaf node, and which are not accompanied by the newSuperior parameter or the **IBMModifyDNRealignDNAttributesControl** control or the **IBMTimeLimitControl** control.
2. Complex Modify DN operations are those which either rename a mid-tree (non-leaf) node, or which are accompanied by the newSuperior parameter, or which are accompanied by either the **IBMModifyDNRealignDNAttributesControl** control or the **IBMTimeLimitControl** control.

Simple Modify DN operations are always accepted by the master server, and are replicated if replicaObjects are present in the TDBM backend where a Modify DN operation is applied.

Complex Modify DN operations are only accepted by the master server, and are propagated to replica servers contingent on a determination that replica servers are compatible server versions. A compatible server version is one known to support for Modify DN operations all features and controls implemented by the z/OS Version 1 Release 4 LDAP server including:

- the **IBMModifyDNRealignDNAttributesControl** control
- the **IBMTimeLimitControl** control
- the *newSuperior* parameter
- rename of non-leaf entries (complex Modify DN operations).

If one or more of these features or controls is not supported by a replica server, all complex Modify DN operations will be refused at the master server.

Periodic validation of compatible server versions in replica servers

Periodic checks are made of replica servers by the master server which are intended to increase the likelihood that complex Modify DN operations will be successfully replicated. Following is a description of the mechanisms used by master servers to do such checking.

The LDAP server must be able to establish a connection to each of the replica servers represented by replicaObjects in a given TDBM backends. When the connection is established to a given replica server, the master server determines if the replica server is at a compatible server version based on a query of the root DSE on that server. If a connection cannot be established to a replica server, it is assumed that the server does not provide the requisite support for replication of Modify DN operations, and complex Modify DN operations are refused at the master server. If a connection is established to a replica server and it is determined that the replica is not at a compatible server version, complex Modify DN operations are refused at the master server. Note that replication of simple Modify DN operations is always permitted, and such operations are always performed at the master server.

During operation of the master server, it may enable or disable processing of complex Modify DN operations, dependent on dynamically changing states of replica servers and of replicaObjects within the master server's TDBM backend. It is possible for the server to refuse complex Modify DN operations after having accepted them for some period of time, and it is possible for the server to accept complex Modify DN operations after having refused them for some period of time. Such a change can be triggered by several events. Each replication cycle tests connections to all replica servers defined by replicaObjects in the TDBM backend, and if a connection can no longer be established to at least one of the replica servers (even if it had been established to the same replica on the previous replication cycle), the master server begins refusing complex Modify DN operations. If all connections succeed but it is determined that one or more of the replica servers is not at a compatible server version (such as might happen, for example, when the replica server has been stopped when running one version of the LDAP server code and subsequently restarted using a different version of the LDAP server code), the master server begins refusing complex Modify DN operations. Only if connections may be established successfully to all replica servers and if they are determined to be running a compatible server version will the master server resume accepting complex Modify DN operations.

Other possible events which may influence whether the master server accepts or refuses complex Modify DN operations involve:

- the addition of new replicaObjects
- deletion of existing replicaObjects
- modification of existing replicaObjects in the TDBM backend.

Each of these causes the master server to temporarily suspend processing of complex Modify DN operations, until the check of replica servers at the start of the next replication cycle, at which point the replica server version levels will be used to determine whether the master server resumes accepting complex Modify DN operations.

Loss of replication synchronization due to incompatible replica server versions

The LDAP Server replication model runs periodically, rather than continuously, and the state of the replica is not checked until the start of each replication cycle. A complex Modify DN operation could be accepted or rejected based on inaccurate information about the state of a replica server between the start of two replication cycles. As a consequence, the replication process could stall and the synchronization between the master server and its replicas could be lost.

Attention

It is highly recommended that before starting a master server containing replica objects which may be the recipient of complex Modify DN operations, that the LDAP server administrator ensure that replica servers are each at a compatible server version level.

This will preclude any situations where replication will become stalled due to the presence of complex Modify DN operations which can not be propagated to all servers in the replica ring. To determine whether a replica server is at a compatible version level, submit a root DSE search to that server, similar to the following:

```
ldapsearch -h ldaphost -p ldapport -v 3 -s base -b "" objectclass=*
```

where *ldaphost* represents the hostname on which the replica server runs and *ldapport* is the port number on which the replica server is listening.

If the attribute type **ibmDirectoryVersion** is returned on the root DSE search, its value must greater than or equal to z/OS V1R4 to be a compatible server version. If the **ibmDirectoryVersion** attribute is not returned on the root DSE search, or if the **ibmDirectoryVersion** attribute is returned on the root DSE search but its value is not greater than or equal to z/OS V1R4, the server is an incompatible server version.

Loss of replication synchronization due to incompatible replica server versions - recovery

If at some point a master server accepts a complex Modify DN operation which can not be replicated, there are several means of recovering from this situation. The best method of recovering from this situation is to ensure that all replica servers are reachable from the master server, and that all replica servers are running at a compatible version level (this may entail stopping some replica servers and restarting them at a compatible version level). Once this state has been reached, queued changes awaiting propagation to replica servers will drain from the queue at the master server and the replication process will resume normal operation.

An alternative is to delete the replicaObject from the master server corresponding to the replica server which is currently unreachable or which is running at an incompatible server level. Note that this will result in loss of synchronization with that replica server, and if one wishes to later restart the offending replica (such as, after it has been brought up to a compatible server version) it will be necessary to take a backup of the master server contents and restore those contents to the replica server before restarting it, to ensure the two directories are synchronized.

Replication of Modify DN operations, shared databases, and compatibility

Since z/OS V1R4, the LDAP Server is backward-compatible with earlier releases of the LDAP Server with respect to replication of Modify DN operations. If an earlier release of the LDAP Server stores replication change records for Modify DN operations in a TDBM backend which is subsequently used by a z/OS V1R4 or later LDAP Server, this higher level of server is capable of replicating Modify DN operations which have not yet been propagated to the replica servers. This backward-compatibility provides for migration to the new version of the z/OS LDAP Server from earlier server versions which support the TDBM backend.

It should be noted that once a z/OS V1R4 or later LDAP Server has performed Modify DN operations (of either simple or complex varieties) and stored replication change records in a given DB2-based TDBM backend, an earlier release of the z/OS LDAP Server should not be run against this backend. Replication change records for Modify DN operations written by the z/OS V1R4 or later LDAP Server are not recognized by earlier versions of the z/OS LDAP Server, and will cause failure of the replication process.

Activity log and console listing of Modify DN operations

The server will make use of the Activity Log (see “Activity Logging” on page 113) and the system console log to record the parameters of all successful Modify DN operations. This will include all values in the operation request, including control values and criticalities. By recording the details of the Modify DN operation request.

To record the Modify DN operation details in the Activity Log, the MSGS level of Activity Log operation must be enabled (see “Activity Logging” on page 113).

It should be noted that while the server is running, it may be necessary to perform a flush of the Activity Log (see “Activity Logging” on page 113) to view all messages for Modify DN operations performed to date.

Regardless of whether the Activity Log is configured to record information about Modify DN operations, the operation details (parameter values and control values) will be written to the system console log unconditionally for all successful complex Modify DN operations. If the customer determines this information is needed for diagnostic purposes within a limited amount of time after the directory data has been changed, it will still be available if archived copies of the system console log are maintained. Following is sample Activity Log output from Modify DN operations, with descriptions of the operation parameters and controls which generated the messages:

Example Modify DN operation submitted to the server:

```
ldapmodrdn -h ldaphost -p ldapport -D adminDN -w passwd  -l 100 -r "o=O1,o=Deltawing,c=AU" "o=O1 Renamed"
```

This operation specifies a time limit of 100 seconds, renames the DN `o=O1,o=Deltawing,c=AU` by modifying the RDN to `o=O1 Renamed`, and deletes the old RDN value from the entry.

Following is the Activity Log output produced by successful completion of this operation:

```
Mon Feb  4 10:36:27 2002 GLD0215I End Successful Modify DN operation: dn =>
 o=O1,o=Deltawing,c=AU; newrdn => o=O1 Renamed; deleteoldrdn => TRUE
Mon Feb  4 10:36:27 2002 GLD0216I End Successful Modify DN operation: dn =>
 o=O1,o=Deltawing,c=AU; newSuperior => Not present in operation request.
Mon Feb  4 10:36:27 2002 GLD0217I End Successful Modify DN operation: dn =>
 o=O1,o=Deltawing,c=AU; control type=> 1.3.18.0.2.10.10; control criticality => TRUE;
 control value (hexadecimal) => x'f1f0f000'
```

Example Modify DN operation submitted to the server:

```
ldapmodrdn -h ldaphost -p ldapport -D adminDN -w passwd -l 300 -a  "ou=Sport,
 ou=Production, ou=Vision On Demand, ou=Delta Home Media  Ltd., o=Deltawing, c=AU"
 "ou=Renamed Sport"
```

This operation specifies a time limit of 300 seconds, requests that DN realignment be performed, and renames the DN `ou=Sport, ou=Production, ou=Vision On Demand, ou=Delta Home Media  Ltd., o=Deltawing, c=AU` by modifying the RDN to `ou=Renamed Sport`.

Following is the Activity Log output produced by successful completion of this operation:

```
Mon Feb  4 09:44:24 2002 GLD0215I End Successful Modify DN operation: dn =>  ou=Sport,
 ou=Production, ou=Vision On Demand, ou=Delta Home Media Ltd., o=Deltawing, c=AU;
 newrdn => ou=Renamed Sport; deleteoldrdn => FALSE
Mon Feb  4 09:44:24 2002 GLD0216I End Successful Modify DN operation: dn =>  ou=Sport,
ou=Production, ou=Vision On Demand, ou=Delta Home Media Ltd., o=Deltawing, c=AU;
 newSuperior => Not present in operation request.
Mon Feb  4 09:44:24 2002 GLD0217I End Successful Modify DN operation: dn =>  ou=Sport,
ou=Production, ou=Vision On Demand, ou=Delta Home Media Ltd., o=Deltawing, c=AU;
control type => 1.3.18.0.2.10.11; control criticality =>  TRUE; control value
(hexadecimal) => x'Not present in operation request.'
```

```
Mon Feb  4 09:44:24 2002 GLD0217I End Successful Modify DN operation: dn =>  ou=Sport,
 ou=Production, ou=Vision On Demand, ou=Delta Home Media Ltd., o=Deltawing, c=AU;
 control type => 1.3.18.0.2.10.10; control criticality =>  TRUE; control value
 (hexadecimal) => x'f3f0f000'
```

Example Modify DN operation submitted to the server:

```
ldapmodrdn -h ldaphost -p ldapport -D adminDN -w passwd -l 10500 -a  -s "" "cn=Steve
 Lawlor, ou=English, ou=Languages, ou=Education Requirements Research, ou=Market Research,
 ou=Home Ed, ou=Delta Home Media Ltd., o=Deltawing,c=AU" "o=TopOfDir"
```

This operation specifies a time limit of 10500 seconds, requests that DN realignment be performed, renames the DN `cn=Steve Lawlor, ou=English, ou=Languages, ou=Education Requirements Research, ou=Market Research, ou=Home Ed, ou=Delta Home Media Ltd., o=Deltawing,c=AU` by modifying the RDN to `o=TopOfDir`, and relocates the entry to the root of the directory using the newSuperior parameter **(-s "")**.

Following is the Activity Log output produced by successful completion of this operation:

```
Mon Feb  4 10:06:20 2002 GLD0215I End Successful Modify DN operation: dn => cn=Steve
 Lawlor, ou=English, ou=Languages, ou=Education Requirements Research, ou=Market
 Research, ou=Home Ed, ou=Delta Home Media Ltd., o=Deltawing, c=AU; newrdn =>
o=TopOfDir; deleteoldrdn => FALSE
Mon Feb  4 10:06:20 2002 GLD0216I End Successful Modify DN operation: dn => cn=Steve
Lawlor, ou=English, ou=Languages, ou=Education Requirements Research, ou=Market
Research, ou=Home Ed, ou=Delta Home Media Ltd., o=Deltawing,c=AU; newSuperior =>
Mon Feb  4 10:06:20 2002 GLD0217I End Successful Modify DN operation: dn => cn=Steve
Lawlor, ou=English, ou=Languages, ou=Education Requirements Research, ou=Market
 Research, ou=Home Ed, ou=Delta Home Media Ltd., o=Deltawing,c=AU; control type =>
1.3.18.0.2.10.11; control criticality => TRUE; control value (hexadecimal) => x'Not
 present in operation request.'
Mon Feb  4 10:06:20 2002 GLD0217I End Successful Modify DN operation: dn => cn=Steve
Lawlor, ou=English, ou=Languages, ou=Education Requirements Research, ou=Market
 Research, ou=Home Ed, ou=Delta Home Media Ltd., o=Deltawing,c=AU; control type =>
 1.3.18.0.2.10.10; control criticality => TRUE; control value (hexadecimal) => x'f1f0f5f0f000'
```

Note: In the sample Activity Log output above, when the newSuperior parameter contains a zero-length string (""), the Activity Log message containing the newSuperior value for the operation is printed as a blank.

Chapter 16. Accessing RACF information

RACF provides definitions of users and groups, as well as access control for resources. The LDAP server can provide LDAP access to the user and group information stored in RACF.

Using SDBM, the RACF database backend of the LDAP server, you can:
- Add new users and groups to RACF
- Add users to groups (connections)
- Modify RACF information for users and groups
- Retrieve RACF information for users and groups
- Delete users and groups from RACF
- Remove users from groups (connections)
- Retrieve RACF user password envelope

The SDBM database of the LDAP server implements portions of the **adduser**, **addgroup**, **altuser**, **altgroup**, **deluser**, **delgroup**, **listuser**, **listgrp**, **connect**, **remove**, and **search** RACF commands. (See "SDBM operational behavior" on page 220 for more information.) An individual user has the same authority through SDBM as with normal RACF commands. The SDBM database of the LDAP server makes use of the R_Admin "run command" interface to accomplish its access to RACF data. As a result, this support is subject to the restrictions of the R_Admin interface. See *z/OS Security Server RACF Callable Services* for more information regarding these restrictions. One restriction in particular affects return of search results. Refer to "RACF restriction on amount of output" on page 227 for more details.

The SDBM database allows for directory authentication (or bind) using the RACF user ID and password. The RACF user ID must have an OMVS segment defined and an OMVS UID present. The RACF user and group information that make up an identity can be used to establish access control on other LDAP directory entities. This expands use of the RACF identity to the rest of the LDAP-managed namespace. Note the following when using RACF access:

- An LDAP simple bind to a z/OS LDAP server using RACF access support but having a non-RACF security manager will succeed as long as the **__passwd()** call made by the LDAP server is successful. However, no group membership information will be available for the bound distinguished name if the security manager is not RACF.
- An LDAP simple bind made to a z/OS LDAP server using RACF access support continues to provide a successful or unsuccessful LDAP return code. In addition, if the LDAP return code being returned is **LDAP_INVALID_CREDENTIALS**, additional information is provided in the "message" portion of the LDAP result. The additional information is an LDAP-unique reason code and reason code text in the following format:

  `Rnnnnnn text`

  The following *errno* values returned by **__passwd()** will have an LDAP reason code defined for them:

Table 30. The errno values returned by _passwd()

errno **value**	**Reason**	**Text**
EMVSEXPIRE	R000100	The password has expired.
EMVSPASSWORD	R000101	The new password is not valid.
EMVSSAFEXTRERR	R000102	The userid has been revoked.
ESRCH	R000104	The password is not correct or the user id is not completely defined (missing password or uid).
EACCES	R000104	The password is not correct or the user id is not completely defined (missing password or uid).
EINVAL	R000105	A bind argument is not valid.

For all of these reason codes, the LDAP return code will be **LDAP_INVALID_CREDENTIALS**.

If the SDBM database is to be used for authentication purposes only, consider having your clients use the **authenticateOnly** server control, to streamline bind processing. This supported control overrides any extended group membership searching and default group membership gathering and is supported for Version 3 clients. See Appendix D, “Supported server controls,” on page 459 for more information.

Note that the SDBM backend only updates the default RACF on a given system. That is, the **AT** and **ONLYAT** clauses of the RACF commands, used to redirect RACF commands, are not exploited by SDBM.

See *z/OS Security Server RACF Command Language Reference* for more information about the supported RACF commands.

See “Setting up for SDBM” on page 44 for information on getting your LDAP server configured with SDBM.

Note: The use of RACF passtickets is supported by the z/OS LDAP server. It is recommended that the LDAP server be run as a started task if RACF passticket support will be used. The job name associated with the LDAP Server started task should be used as the application name when generating RACF passtickets. Refer to *z/OS Security Server RACF Macros and Interfaces* for more information about RACF passtickets.

Mapping LDAP-style names to RACF attributes

Following are tables that show the RACF attribute name and the corresponding LDAP-style attribute name for user (Table 31), group (Table 32 on page 217), and connection (Table 33 on page 218). Not all attribute names apply to all versions.

Table 31. Mapping of LDAP-style names to RACF attributes (user)

RACF segment name	RACF attribute name in altuser/adduser string	LDAP-style attribute name
User base or Group base	OWNER	racfOwner
User base or Group base	DATA	racfInstallationData
User base or Group base	MODEL	racfDatasetModel
User base or Group base	Not modifiable; listuser/listgrp displays as CREATED	racfAuthorizationDate
User base	Multi-value: ADSP, SPECIAL, OPERATIONS, GRPACC, AUDITOR, OIDCARD, UAUDIT, or any other one-word values, such as NOEXPIRED and NOOMVS	racfAttributes
User base	PASSWORD	racfPassword
User base	password envelope — not modifiable	racfPasswordEnvelope
User base	Not modifiable - displayed as PASS-INTERVAL	racfPasswordInterval
User base	Not modifiable - displayed as PASSDATE	racfPasswordChangeDate
User base	NAME	racfProgrammerName
User base	DFLTGRP	racfDefaultGroup

Table 31. Mapping of LDAP-style names to RACF attributes (user) (continued)

RACF segment name	RACF attribute name in altuser/adduser string	LDAP-style attribute name
User base	Not modifiable - displayed as LAST-ACCESS	racfLastAccess
User base	SECLEVEL	racfSecurityLevel
User base	ADDCATEGORY	racfSecurityCategoryList
User base	REVOKE	racfRevokeDate
User base	RESUME	racfResumeDate
User base	WHEN(DAYS())	racfLogonDays
User base	WHEN(TIME())	racfLogonTime
User base	CLAUTH	racfClassName
User base	GROUP	racfConnectGroupName
User base	AUTH not displayed by LDAP	racfConnectGroupAuthority
User base	UACC not displayed by LDAP	racfConnectGroupUACC
User base	SECLABEL	racfSecurityLabel
DFP segment - common to group or user	DATAAPPL	SAFDfpDataApplication
DFP segment - common to group or user	DATACLAS	SAFDfpDataClass
DFP segment - common to group or user	MGMTCLAS	SAFDfpManagementClass
DFP segment - common to group or user	STORCLAS	SAFDfpStorageClass
TSO segment	ACCTNUM	SAFAccountNumber
TSO segment	COMMAND	SAFDefaultCommand
TSO segment	DEST	SAFDestination
TSO segment	HOLDCLASS	SAFHoldClass
TSO segment	JOBCLASS	SAFJobClass
TSO segment	MSGCLASS	SAFMessageClass
TSO segment		SAFDefaultLoginProc
TSO segment	SIZE	SAFLogonSize
TSO segment	MAXSIZE	SAFMaximumRegionSize
TSO segment	SYSOUTCLASS	SAFDefaultSysoutClass
TSO segment	USERDATA	SAFUserdata
TSO segment	UNIT	SAFDefaultUnit
TSO segment	SECLABEL	SAFTsoSecurityLabel
LANGUAGE segment	PRIMARY	racfPrimaryLanguage
LANGUAGE segment	SECONDARY	racfSecondaryLanguage
CICS®® segment	OPIDENT	racfOperatorIdentification
CICS segment	OPCLASS	racfOperatorClass
CICS segment	OPPRTY	racfOperatorPriority
CICS segment	XRFSOFF	racfOperatorReSignon
CICS segment	TIMEOUT	racfTerminalTimeout

Table 31. Mapping of LDAP-style names to RACF attributes (user) (continued)

RACF segment name	RACF attribute name in altuser/adduser string	LDAP-style attribute name
CICS segment	RSLKEY	racfRslKey
CICS segment	TSLKEY	racfTslKey
OPERPARM segment	STORAGE	racfStorageKeyword
OPERPARM segment	AUTH	racfAuthKeyword
OPERPARM segment	MFORM	racfMformKeyword
OPERPARM segment	LEVEL	racfLevelKeyword
OPERPARM segment	MONITOR	racfMonitorKeyword
OPERPARM segment	ROUTCODE	racfRoutcodeKeyword
OPERPARM segment	LOGCMDRESP	racfLogCommandResponseKeyword
OPERPARM segment	MIGID	racfMGIDKeyword
OPERPARM segment	DOM	racfDOMKeyword
OPERPARM segment	KEY	racfKEYKeyword
OPERPARM segment	CMDSYS	racfCMDSYSKeyword
OPERPARM segment	UD	racfUDKeyword
OPERPARM segment	MSCOPE	racfMscopeSystems
OPERPARM segment	ALTGRP	racfAltGroupKeyword
OPERPARM segment	AUTO	racfAutoKeyword
WORKATTR segment	WANAME	racfWorkAttrUserName
WORKATTR segment	WABLDG	racfBuilding
WORKATTR segment	WADEPT	racfDepartment
WORKATTR segment	WAROOM	racfRoom
WORKATTR segment	WAADDR1	racfAddressLine1
WORKATTR segment	WAADDR2	racfAddressLine2
WORKATTR segment	WAADDR3	racfAddressLine3
WORKATTR segment	WAADDR4	racfAddressLine4
WORKATTR segment	WAACCNT	racfWorkAttrAccountNumber
User OMVS segment	UID	racfOmvsUid
User OMVS segment	SHARED, AUTOUID	racfOmvsUidKeyword
HOME		racfOmvsHome
User OMVS segment	PROGRAM	racfOmvsInitialProgram
User OMVS segment	CPUTIMEMAX	racfOmvsMaximumCPUTime
User OMVS segment	ASSIZEMAX	racfOmvsMaximumAddressSpaceSize
User OMVS segment	FILEPROCMAX	racfOmvsMaximumFilesPerProcess
User OMVS segment	PROCUSERMAX	racfOmvsMaximumProcessesPerUID
User OMVS segment	THREADSMAX	racfOmvsMaximumThreadsPerProcess
User OMVS segment	MMAPAREAMAX	racfOmvsMaximumMemoryMapArea
User OMVS segment	MEMLIMIT	racfOmvsMemoryLimit
User OMVS segment	SHMEMMAX	racfOmvsSharedMemoryMaximum
Netview segment	IC	racfNetviewInitialCommand

Table 31. Mapping of LDAP-style names to RACF attributes (user) (continued)

RACF segment name	RACF attribute name in altuser/adduser string	LDAP-style attribute name
Netview segment	CONSNAME	racfDefaultConsoleName
Netview segment	CTL	racfCTLKeyword
Netview segment	MSGRECVR	racfMessageReceiverKeyword
Netview segment	OPCLASS	racfNetviewOperatorClass
Netview segment	DOMAINS	racfDomains
Netview segment	NGMFADMN	racfNGMFADMKeyword
DCE segment	UUID	racfDCEUUID
DCE segment	DCENAME	racfDCEPrincipal
DCE segment	HOMECELL	racfDCEHomeCell
DCE segment	HOMEUUID	racfDCEHomeCellUUID
DCE segment	AUTOLOGIN	racfDCEAutoLogin
User OVM segment	UID	racfOvmUid
User OVM segment	HOME	racfOvmHome
User OVM segment	PROGRAM	racfOvmInitialProgram
User OVM segment	FSROOT	racfOvmFileSystemRoot
LNOTES segment	SNAME	racfLNotesShortName
NDS segment	UNAME	racfNDSUserName
KERB segment	KERBNAME	krbPrincipalName
KERB segment	MAXTKTLFE	maxTicketAge
KERB segment	Not modifiable - displayed as KEY VERSION	racfCurKeyVersion
KERB segment	ENCRYPT	racfEncryptType
PROXY segment	BINDDN	racfLDAPBindDN
PROXY segment	BINDPW - value displayed is YES or NO	racfLDAPBindPw
PROXY segment	LDAPHOST	racfLDAPHost
EIM segment	LDAPPROF	racfLDAPProf

Table 32. Mapping of LDAP-style names to RACF attributes (group)

RACF segment name	RACF attribute name in altgroup/addgroup string	LDAP-style attribute name
User base or Group base	OWNER	racfOwner
User base or Group base	DATA	racfInstallationData
User base or Group base	MODEL	racfDatasetModel
User base or Group base	Not modifiable; listuser/listgrp displays as CREATED	racfAuthorizationDate
Group base	SUPGROUP	racfSuperiorGroup
Group base	TERMUACC	racfGroupNoTermUAC
Group base	UNIVERSAL	racfGroupUniversal
Group base	Not modifiable - listgrp displays as SUBGROUP(S)	racfSubGroupName

Table 32. Mapping of LDAP-style names to RACF attributes (group) (continued)

RACF segment name	RACF attribute name in altgroup/addgroup string	LDAP-style attribute name
Group base	Not modifiable - listgrp displays as USER(S)	racfGroupUserids
Group OMVS segment	GID	racfOmvsGroupId
Group OMVS segment	SHARED, AUTOGID	racfOmvsGroupIdKeyword
Group OVM segment	GID	racfOvmGroupId
DFP segment - common to group or user	DATAAPPL	SAFDfpDataApplication
DFP segment - common to group or user	DATACLAS	SAFDfpDataClass
DFP segment - common to group or user	MGMTCLAS	SAFDfpManagementClass
DFP segment - common to group or user	STORCLAS	SAFDfpStorageClass

Table 33. Mapping of LDAP-style names to RACF attributes (connection)

RACF segment name	RACF attribute name in connect string	LDAP-style attribute name
Connection base	Multi-value: ADSP, AUDITOR GRPACC, OPERATIONS, SPECIAL	racfConnectAttributes
Connection base	AUTHORITY	racfConnectGroupAuthority
Connection base	Not modifiable - displayed as CONNECT-DATE	racfConnectAuthDate
Connection base	Not modifiable - displayed as CONNECTS	racfConnectCount
Connection base	Not modifiable - displayed as LAST-CONNECT	racfConnectLastConnect
Connection base	OWNER	racfConnectOwner
Connection base	RESUME	racfConnectResumeDate
Connection base	REVOKE	racfConnectRevokeDate
Connection base	UACC	racfConnectGroupUACC

Special usage of racfAttributes and racfConnectAttributes

The **racfAttributes** attribute is a multi-valued attribute that can be used to specify any single-word keywords that can be specified on a RACF **adduser** or **altuser** command. For example, **racfAttributes** can be used to add a RACF user entry with 'ADSP GRPACC NOPASSWORD' or modify a RACF user entry with 'NOGRPACC SPECIAL NOEXPIRED RESUME NOOMVS' .

Similarly, **racfConnectAttributes** can be used to specify any single-word keywords that can be specified on a RACF **connect** command.

RACF namespace entries

When the SDBM database is used to make RACF information accessible over the LDAP protocol, the top four entries in the hierarchy are reserved, read-only, and generated by the server. The purpose of these reserved entries is to enable a hierarchical representation of RACF users, groups, and connections. Note that the root of the RACF namespace can be any DN and is not required to contain the **sysplex** attribute as the RDN™. For example, the top four entries in Figure 34 on page 219 are:

- `cn=RACFA,o=IBM,c=US` (suffixDN)
- `profileType=User,cn=RACFA,o=IBM,c=US`
- `profileType=Group,cn=RACFA,o=IBM,c=US`

- `profileType=Connect,cn=RACFA,o=IBM,c=US`

The value of the top DN is generated from the suffix line in the **slapd.conf** file for the SDBM database entry (see “Setting up for SDBM” on page 44).

Following is a high-level diagram of the RACF backend.

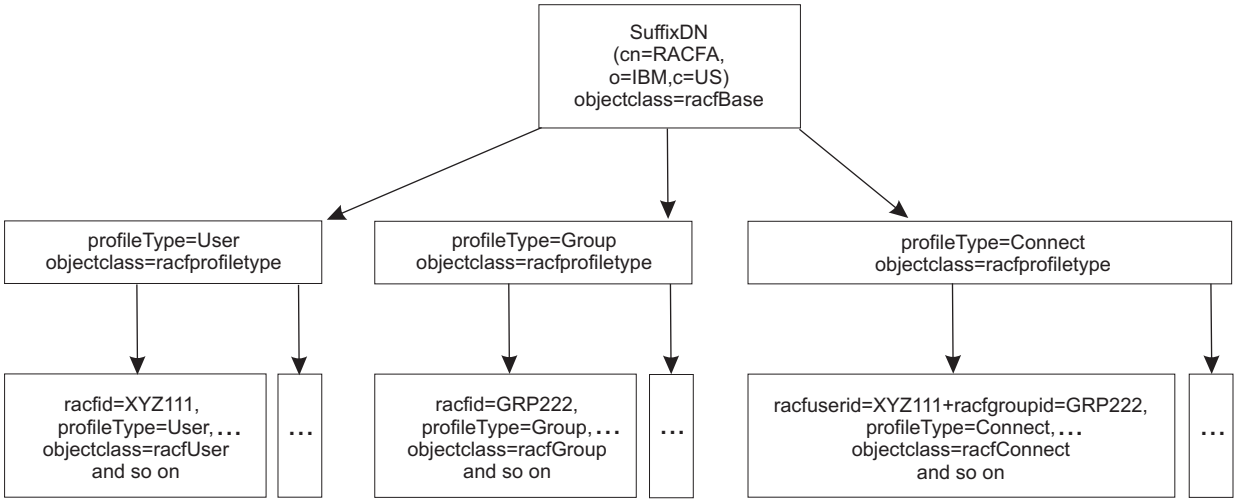


Figure 34. RACF namespace hierarchy

SDBM schema information

SDBM has an internal schema that it uses to check entries during add. The schema contains all the attributes and object classes that were in the **slapd.at.racf** and **slapd.oc.racf** schema files shipped in previous releases. The schema cannot be modified. The DN of the SDBM schema entry generated by the server is `cn=schema,`*`suffixDN`*. For example, if *suffixDN* is `cn=RACFA,o=IBM,c=US` then the schema entry DN would be

```
cn=schema,cn=RACFA,o=IBM,c=US
```

In this case, the SDBM schema can be published (displayed) by the following search command:

```
ldapsearch -h ldaphost -p ldapport -b "cn=SCHEMA,cn=RACFA,o=IBM,c=US" "objectclass=subschema"
```

See Appendix B, “SDBM schema,” on page 443 for the complete SDBM schema.

When running TDBM and SDBM together, the TDBM schema will be used for initial DN normalization for selecting the appropriate backend. Since TDBM schema will be used for DN normalization, all schema elements used in formulating distinguished names for SDBM must appear in TDBM’s schema definition.

SDBM support for pound sign

An SDBM DN can contain a pound sign (#) anywhere in the DN, including the suffix. The pound sign must be escaped by preceding it with a single backslash (\). Note that the suffix in the configuration file must use two backslashes (\\) to escape a pound sign, but only a single backslash is used in a DN.

For example, if the SDBM suffix in the configuration file is

```
suffix sysplex=my\\#1plex
```

then the DN for the RACF user ab#c would be

```
racfid=ab\#c,profiletype=user,sysplex=my\#1plex
```

Pound signs in DN values returned by SDBM from a search are always escaped by a single backslash.

When specifying a value containing a pound sign for an attribute within an add or modify request or within a search filter, do **not** escape the pound sign with a backslash, even if the value is a DN. For instance, to search for all RACF users starting with `user#`, use the search filter `racfid=user#*`. To add a user with default group `d#1grp`, specify:

```
racfdefaultgroup: racfid=d#1grp,profiletype=group,sysplex=my#1plex
```

within the entry.

SDBM operational behavior

Table 34 shows how the SDBM database behaves during different LDAP operations.

Table 34. RACF backend behavior

Target DN	LDAP operation behavior
suffixDN	**Add** Error: Unwilling to perform **Modify** Error: Unwilling to perform **Delete** Error: Unwilling to perform **Modify DN** Error: Unwilling to perform **Compare** Compare attribute **Search base** Return requested attributes **Search one level** Perform a base search against each subordinate of this entry **Search subtree** See “Searching the entire RACF database” on page 226 **Bind** Error: No credentials
profiletype=User,suffixDN	**Add** Error: Unwilling to perform **Modify** Error: Unwilling to perform **Delete** Error: Unwilling to perform **Modify DN** Error: Unwilling to perform **Compare** Compare attribute **Search base** Return requested attributes **Search one level** See “Searching the entire RACF database” on page 226 **Search subtree** See “Searching the entire RACF database” on page 226 **Bind** Error: No credentials

Table 34. RACF backend behavior (continued)

Target DN	LDAP operation behavior
profiletype=Group,suffixDN	**Add** Error: Unwilling to perform **Modify** Error: Unwilling to perform **Delete** Error: Unwilling to perform **Modify DN** Error: Unwilling to perform **Compare** Compare attribute **Search base** Return requested attributes **Search one level** See “Searching the entire RACF database” on page 226 **Search subtree** See “Searching the entire RACF database” on page 226 **Bind** Error: No credentials
profiletype=Connect,suffixDN	**Add** Error: Unwilling to perform **Modify** Error: Unwilling to perform **Delete** Error: Unwilling to perform **Modify DN** Error: Unwilling to perform **Compare** Compare attribute **Search base** Return requested attributes **Search one level** See “Searching the entire RACF database” on page 226 **Search subtree** See “Searching the entire RACF database” on page 226 **Bind** Error: No credentials

Table 34. RACF backend behavior (continued)

Target DN	LDAP operation behavior
racfid=XYZ111,profiletype=User, suffixDN	**Add** Perform an **adduser** RACF command using USER=XYZ111 **Modify** Perform an **altuser** RACF command using USER=XYZ111 **Delete** Perform a **deluser** RACF command using USER= XYZ111 **Modify DN** Error: Unwilling to perform **Compare** Compare requested attribute with data returned from **listuser** RACF command using USER=XYZ111 **Search base** Perform a **listuser** RACF command using USER=XYZ111 **Search one level** Empty search results (this is a leaf node in the hierarchy) **Search subtree** Perform a **listuser** RACF command using USER=XYZ111 **Bind** If bind type is not simple, error: Unwilling to perform, else use **__passwd()** to verify the user ID and password combination and perform a **listuser** RACF command using USER=XYZ111
racfid=GRP222,profiletype= Group, suffixDN	**Add** Perform an **addgroup** RACF command using GROUP=GRP222 **Modify** Perform an **altgroup** RACF command using GROUP=GRP222 **Delete** Perform a **delgroup** RACF command using GROUP=GRP222 **Modify DN** Error: Unwilling to perform **Compare** Compare requested attribute with data returned from **listgrp** RACF command using GROUP=GRP222 **Search base** Perform a **listgrp** RACF command using GROUP=GRP222 **Search one level** Empty search results (this is a leaf node in the hierarchy) **Search subtree** Perform a **listgrp** RACF command using GROUP=GRP222 **Bind** Error: No credentials

Table 34. RACF backend behavior (continued)

Target DN	LDAP operation behavior
racfuserid=XYZ111+racfgroupid=GRP222,profiletype=Connect, suffixDN	**Add** Perform a **connect** RACF command for USER=XYZ111 using GROUP=GRP222 **Modify** Perform a **connect** RACF command for USER=XYZ111 using GROUP=GRP222 **Delete** Perform a **remove** RACF command for USER=XYZ111 using GROUP=GRP222 **Modify DN** Error: Unwilling to perform **Compare** Compare requested attribute with data returned from **listuser** RACF command using USER=XYZ111 **Search base** Perform a **listuser** RACF command using USER=XYZ111 **Search one level** Empty search results (this is a leaf node in the hierarchy) **Search subtree** Perform a **listuser** RACF command using USER=XYZ111 **Bind** Error: No credentials

If LDAP is running with an SDBM backend, the **ldap_modify** and **ldap_add** APIs can return **LDAP_OTHER** or **LDAP_SUCCESS** and have completed a partial update to an entry in RACF. The results will match what would occur if the update were done using the RACF **altuser**, **altgroup**, and **connect** commands. If several RACF attributes are being updated and one of them is in error, RACF might still update the other attributes, without, in some cases, returning an error message. If there is a RACF message, LDAP always returns it in the result

The RACF **connect** command is used to both add a user connection to a group and to modify a user's connection to a group. As a result, the SDBM add and modify support for connection entries is different than normal LDAP support:

- When adding a connection entry that already exists, the entry is modified using the specified attributes. There is no indication returned that the entry already existed.
- When modifying a connection entry that does not exist, the entry is added using the specified attributes. There is no indication returned that the entry did not exist.

Notes about specifying attribute values:

1. In LDAP, the format of the value of the Kerberos principal name attribute, **krbPrincipalName**, is *userid@realm*. In SDBM, the *userid* portion of the name is case-sensitive while the *realm* portion of the name is not. In addition, SDBM only operates on the RACF local realm. If the realm specified in the value is not the local realm, the operation fails. For example, when searching in SDBM for a user entry using the search filter `krbPrincipalName=krbuser1@myrealm.com`, the search fails if `myrealm.com` is not the RACF local realm.

   To facilitate using the **krbPrincipalName** attribute, the attribute value can be specified without the *@realm* portion. In this case, the realm is assumed to be the RACF local realm. For example, when adding a user entry with `krbuser1` as the *userid* portion of that Kerberos principal name, the **krbPrincipalName** attribute can be specified as

   ```
   krbPrincipalName=krbuser1
   ```

   or

```
krbPrincipalName=krbuser1@myrealm.com
```

where `myrealm.com` is the RACF local realm.

The **krbPrincipalName** value returned by SDBM from a search is always the complete principal name, *userid@realm*, where *realm* is the RACF local realm.

2. There are several SDBM attributes whose value is a RACF user or group name. For convenience, this value can be specified either as just the RACF name or as the complete LDAP DN. For example, when adding a user with a default group of `grp222`, the **racfDefaultGroup** attribute can be specified as

```
racfDefaultGroup=grp222
```

or

```
racfDefaultGroup=racfid=grp222,profiletype=group,sysplex=myplex
```

where `sysplex=myplex` is the SDBM suffix.

The value returned by SDBM from a search is always the complete LDAP DN.

3. For multi-value attributes, the RACF **altuser** command does not always support the ability to both add a value and replace the existing value. As a result, SDBM does not always respect the type of modification (add versus replace) that is specified in a modify command. Values for the following multi-value attributes are always added to the existing value (even if replace is specified): **racfAttributes**, **racfClassName**, **racfConnectAttributes**, **racfLevelKeyword**, **racfMformKeyword**, **racfMonitorKeyword**, **racfRslKey**, **racfSecurityCategoryList**, **racfTslKey** . Values for the following multi-value attributes always replace the existing value (even if add is specified): **racfDomains**, **racfMscopeSystems**, **racfNetviewOperatorClass**, **racfOperatorClass**, **racfRoutcodeKeyword** . Values for the following multi-value attributes either are added to the existing values or replace the existing values, depending on the new and existing values: **racfAuthKeyword**.
4. In order to update CICS-related attributes, CICS must be set up on your system; otherwise, errors result.
5. For modify, if a request is made to delete a specific attribute value for an attribute where specific values cannot be selectively deleted, **LDAP_UNWILLING_TO_PERFORM** is returned. There are three attributes where specific attribute values are accepted: **racfAttributes**, **racfSecurityCategoryList**, and **racfConnectAttributes**. If an attempt is made to delete any attribute that has no corresponding delete command in RACF, **LDAP_UNWILLING_TO_PERFORM** is returned.

Mapping SDBM requests to RACF commands

SDBM uses the RACF R_Admin interface to issue RACF commands to process LDAP requests. The RACF commands are issued under the identity of the bind user. See *z/OS Security Server RACF Command Language Reference* for more information on the authorization required to use each RACF command. Following is a mapping of the RACF commands issued by SDBM for each type of LDAP request.

1. Add, modify, delete, compare, and bind an entry for user U result in **adduser u**, **altuser u**, **deluser u**, **listuser u**, and **listuser u** respectively.
2. Add, modify, delete, and compare an entry for group G result in **addgroup g**, **altgroup g**, **delgroup g**, and **listgrp g** respectively.
3. Add, modify, delete, compare an entry for the connection of user U to group G result in **connect u group(g)**, **connect u group(g)**, **remove u group(g)**, and **listuser u** respectively.
4. Search:
   a. Searching for a user entry results in either **listuser** *value*, **search class(user) filter**(*value*), **getpwuid**(*value*), or **search class(user) uid**(*value*)
   b. Searching for a group entry results in either **listgrp** *value*, **search class(group) filter**(*value*), **getgrgid**(*value*), or **search class(user) gid**(*value*)
   c. Searching for a connect entry results in **listuser** *value*, or **listgrp** *value*, or **search class(group) filter**(*value*) followed by **listgrp** *group* for each *group* returned by **search**

where *value* is the value specified in the search filter.

SDBM search capabilities

SDBM supports a limited set of search filters. Following is a list of the filters and the type of entry that matches each filter:

Table 35. SDBM-supported search filters

Filter	Matching entries
objectclass=*	All entries for RACF users, groups, and connections
racfid=<*any_value*>	User or group entries for the RACF users and groups with the specified name (can contain wildcards)
racflnotesshortname=<*any_value*>	User entry for the RACF user with the specified LNOTES SNAME
racfndsusername=<*any_value*>	User entry for the RACF user with the specified NDS UNAME
racfomvsuid=<*number*>	User entry for a RACF user with the specified OMVS UID
racfomvsuid;allOMVSids=<*number*>	User entries for all the RACF users with the specified OMVS UID
racfomvsgroupid=<*number*>	Group entry for a RACF group with the specified OMVS GID
racfomvsgoupid;allOMVSids=<*number*>	Group entries for all the RACF groups with the specified OMVS GID
krbprincipalname=<*any_name*>	User entry for the RACF user with the specified KERB KERBNAME
racfuserid=<*any_value*>	Connection entries for the RACF users with the specified name (can contain wildcards)
racfgroupid=<*any_value*>	Connection entries to the RACF groups with the specified name (can contain wildcards)
(&(racfuserid=<*any_value*>)(racfgroupid=<*any_value*>))	Connection entries for the RACF users with the first specified name to the RACF groups with the second specified name (both can contain wildcards)

Table 36 shows the search filters that are supported for each search base.

Table 36. RACF backend search filters

Search base	Filters supported
suffixDN (root of the directory)	objectclass=* racfid=<*any_value*> racflnotesshortname=<*any_value*> racfndsusername=<*any_value*> racfomvsuid=<*number*> racfomvsuid;allOMVSids=<*number*> racfomvsgroupid=<*number*> racfomvsgroupid;allOMVSids=<*number*> krbprincipalname=<*any_name*> racfuserid=<*any_value*> racfgroupid=<*any_value*> (&(racfuserid=<*any_value*>)(racfgroupid=<*any_value*>))

Table 36. RACF backend search filters (continued)

Search base	Filters supported
`profileType=user,suffixDN`	objectclass=* racfid=<*any_value*> racflnotesshortname=<*any_value*> racfndsusername=<*any_value*> racfomvsuid=<*number*> racfomvsuid;allOMVSids=<*number*> krbprincipalname=<*any_value*>
`profileType=group,suffixDN`	objectclass=* racfid=<*any_value*> racfomvsgroupid=<*number*> racfomvsgroupid;allOMVSids=<*number*>
`profileType=connect,suffixDN`	objectclass=* racfuserid=<*any_value*> racfgroupid=<*any_value*> (&(racfuserid=<*any_value*>)(racfgroupid=<*any_value*>))
`racfid=abc,profileType=user,suffixDN`	objectclass=*
`racfid=xyz,profileType=group,suffixDN`	objectclass=*
`racfuserid=abc+racfgroupid=xyz,profileType=connect,suffixDN`	objectclass=*

Except for the AND filter for connections, complex search filters that include NOT, AND, OR, Language Environment, or GE constructs are not supported.

The values for the **racfid**, **racfuserid**, and **racfgroupid** filters can include the wild cards supported by RACF. These wild cards are '*' which represents any number of characters, and '%' which represents one character. For example:

```
(&(racfuserid=usr*)(racfgroupid=*grp))
```

searches for all the connections between users whose names begin with `usr` and groups whose names end with `grp`.

Note about searching universal groups: Most of the members of a RACF universal group are not actually contained in the group's list of members. As a result, a search of the entry for a universal group does not return most of the group's members. In addition, a search for the connection entry corresponding to a member of a universal group can return different results depending on the connection search filter that is used:

- If the **racfuserid** part of the connection search filter does not contain a wild card, then the connection entry is returned for the specified **racfuserid**.
- If the **racfuserid** part of the connection search filter contains a wild card, then the connection entry for a user is returned only if the user is explicitly contained in the universal group's list of members.

Searching the entire RACF database

Most searches that query the entire RACF database, for example, a subtree search from one of the top four directory entries, return only the DN (distinguished name) attribute. You may then obtain more specific data about a particular user or group on a follow-up search using a specific DN as the search base. See "RACF-style distinguished names" on page 162 for more information.

The exceptions to this are searches using the "application ID" filters:

```
racflnotesshortname=<any_value>
racfndsusername=<any_value>
krbprincipalname=<any_name>
racfomvsuid=<number>
racfomvsgroupid=<number>
```

Because these searches can match only a single RACF user, the entire user entry is returned in the search results.

RACF restriction on amount of output: When processing LDAP bind, compare, and search requests, SDBM uses the RACF R_Admin interface to issue RACF **search**, **listuser**, and **listgrp** commands. R_Admin limits the number of records in its output to 4096. This means that the RACF **search** command output may be incomplete if you have a large number of users, groups, and connections. Similarly, the RACF **listuser** and **listgrp** output may be incomplete for a user who is a member of many groups and for a group which has many members. If this occurs, the entries returned by SDBM will also be incomplete. In particular, when binding with such a user, the list of groups associated with the user will be incomplete.

Retrieving RACF user password envelope

SDBM returns the RACF user password envelope when the **racfPasswordEnvelope** attribute is specified in the attributes to be returned from a search of a RACF user. The envelope is returned by the LDAP server as a binary data berval (binary data and length). If the **racfPasswordEnvelope** attribute is not specified on the search request, the RACF password envelope is not returned.

Note: When using a utility such as ldapsearch to retrieve the password envelope, the returned value is base-64 encoded.

Using SDBM to change a user password in RACF

There are two ways to use SDBM to change a user password in RACF.

1. The user password of the bind user can be changed during an LDAP simple bind to SDBM. The simple bind can occur as part of an LDAP function such as search, add, modify, or delete. The password change is provided in the password portion of the LDAP simple bind. The password must be in the following format:

   ```
   password/newpassword
   ```

   The forward slash ('/') is used as the indication of a password change during the LDAP simple bind. Password changes made using the LDAP simple bind to the SDBM backend of the z/OS LDAP server are subject to the system password rules. A password change will fail with LDAP return code **LDAP_INVALID_CREDENTIALS** and LDAP reason code of:

   ```
   R000101  The new password is not valid.
   ```

   if the new password does not pass the rules established on the system.

   Note that once the bind succeeds, the password is changed even if the LDAP function eventually fails.

   For example, the following command changes the password for RACF user `U1` from `abc` to `xyz`, assuming the SDBM suffix is `sysplex=sysplexa`:

   ```
   ldapsearch -h ldaphost -p ldapport -D racfid=u1,profiletype=user,sysplex=sysplexa
     -w abc/xyz -s base -V 3 -b "" objectclass=*
   ```

2. To change any user's password, create an LDIF file that modifies the **racfPassword** attribute for that user and then invoke **ldapmodify** to change the password. If the syntax of the new password is not valid, the command fails, returning "ldap_modify: Unknown error". (Note that this response can also be returned under other circumstances.)

   For example, the following LDIF file, `pw.mod`, resets the password for RACF user `U1` to `xyz`, assuming the SDBM suffix is `sysplex=sysplexa`. The `racfAttributes: noexpired` record is added to result in a new password that is not expired. If **noexpired** is not specified, then the password is reset but is expired, requiring `U1` to change the password at next logon.

```
dn: racfid=u1,profiletype=USER,sysplex=sysplexa
changetype: modify
add: x
racfpassword: xyz
racfattributes: noexpired
```

Then, assuming that the RACF user admin1 has the necessary RACF authorization to update RACF, the command:

```
ldapmodify -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,sysplex=sysplexa
  -w passwd -f pw.mod
```

modifies the password for U1.

Using LDAP operation utilities with SDBM

The LDAP operation utilities described in the *z/OS Integrated Security Services LDAP Client Programming* can be used to update data in RACF. Following are some examples. These examples assume that the RACF user admin1 has the necessary RACF authorization to make these RACF updates and that sysplex=sysplexa is the SDBM suffix.

Example: displaying the SDBM schema

To see the contents of the schema used by SDBM, the following search command can be performed:

```
ldapsearch -h ldaphost -p ldapport -b "cn=schema,sysplex=sysplexa" "objectclass=subschema"
```

The beginning of the results of the search will look like:

```
CN=SCHEMA,sysplex=sysplexa
cn=SCHEMA
subtreespecification=NULL
attributetypes=( 2.5.21.5 NAME 'attributeTypes' EQUALITY objectIdentifierFirstComponentMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.3 USAGE directoryOperation )
```

See Appendix B, “SDBM schema,” on page 443 for the complete SDBM schema.

Example: adding a user to RACF

If the LDIF file user.add contains:

```
dn: racfid=newuser,profiletype=user,sysplex=sysplexa
objectclass: racfUser
racfid: newuser
```

The following command will add user ID newuser to RACF:

```
ldapadd -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,sysplex=sysplexa -w
passwd -f user.add
```

Note that the only required attribute to add a user is the user ID specified as **racfid**. This mimics the RACF **adduser** command.

Example: modifying a user in RACF

To add a TSO segment for newuser, the LDIF file user.mods could contain:

```
dn: racfid=newuser,profiletype=user,sysplex=sysplexa
changetype: modify
objectclass: SAFTsoSegment
SAFAccountNumber: 123
SAFHoldClass: H
SAFLogonSize: 1024
```

The command:

```
ldapmodify -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,sysplex=sysplexa -w
passwd -f user.mods
```

modifies the RACF user profile for user ID newuser, adding a TSO segment with the specified values.

Example: searching for user information in RACF

To see the information in RACF for newuser, the following search command can be performed:

```
ldapsearch -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,sysplex=sysplexa -w passwd
 -b "racfid=newuser,profiletype=user,sysplex=sysplexa" "objectclass=*"
```

The results that are returned are most of the data that RACF displays on a **listuser** command, but using LDAP-style attribute names. Following is an example for newuser:

```
racfid=NEWUSER,profiletype=USER,sysplex=sysplexa
objectclass=racfUser
objectclass=racfBaseCommon
objectclass=SAFTsoSegment
racfid=NEWUSER
racfprogrammername=UNKNOWN
racfowner=racfid=ADMIN1,profiletype=USER,sysplex=sysplexa
racfauthorizationdate=98.001
racfdefaultgroup=racfid=GROUPID,profiletype=GROUP,sysplex=sysplexa
racfpasswordchangedata=00.000
racfpasswordinterval=90
racfattributes=NONE
racfrevokedata=NONE
racfresumedate=NONE
racflastaccess=UNKNOWN
racfclassname=NONE
racfinstallationdata=NO-INSTALLATION-DATA
racfdatasetmodel=NO-MODEL-NAME
racflogondays=ANYDAY
racflogontime=ANYTIME
racfconnectgroupname=racfid=GROUPID,profiletype=GROUP,sysplex=sysplexa
racfsecuritylevel=NONE SPECIFIED
racfsecuritycategorylist=NONE SPECIFIED
racfsecuritylabel=NONE SPECIFIED
safaccountnumber=123
safholdclass=H
saflogonsize=00001024
safmaximumregionsize=00000000
safuserdata=0000
```

Example: searching for a user's password envelope in RACF

The following search returns the **racfPasswordEnvelope** attribute:

```
 ldapsearch -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,sysplex=sysplexa  -w passwd -L
 -b racfid=newuser,profiletype=user,sysplex=sysplexa "objectclass=*" racfpasswordenvelope
```

The result returned is:

```
dn: racfid=newuser,profiletype=user,sysplex=sysplexa
racfpasswordenvelope:: base-64_encoded_password_envelope
```

Example: adding a group to RACF

If the LDIF file group.add contains:

```
dn: racfid=grp222,profiletype=group,sysplex=sysplexa
objectclass: racfGroup
racfid: grp222
```

The following command adds group ID grp222 to RACF:

```
ldapadd -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,sysplex=sysplexa -w passwd
  -f group.add
```

Note that the only required attribute to add a group is the group ID specified as racfid. This mimics the RACF **addgroup** command.

The commands for modifying, searching, and removing a RACF group using SDBM are very similar to the corresponding commands for a RACF user. See the examples in this section for a RACF user for more information.

Example: connecting a user to a group in RACF

To connect newuser to group grp222, the LDIF file connect.add could contain:

```
dn: racfuserid=newuser+racfgroupid=grp222,profiletype=connect,sysplex=sysplexa
objectclass: racfconnect
racfuserid: newuser
racfgroupid: grp222
```

The command:

```
ldapadd -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,sysplex=sysplexa -w passwd
  -f connect.add
```

makes newuser a member of the grp222 group. Note that grp222 must be an existing RACF group ID, newuser must be an existing RACF user ID, and the only required attributes to add a connection are **racfuserid** (the user ID) and **racfgroupid** (the group ID).

Example: searching for information about a user's connection to a group in RACF

To see information about newuser's connection to the grp222 group, the following search can be performed:

```
ldapsearch -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,sysplex=sysplexa -w passwd
  -b "racfuserid=newuser+racfgroupid=grp222,profiletype=connect,sysplex=sysplexa" "objectclass=*"
```

The result returned is the information from the GROUP section that RACF displays on a **listuser** command, but using LDAP-style attribute names. Following is an example for newuser's connection to grp222:

```
racfuserid=NEWUSER+racfgroupid=GRP222,profiletype=CONNECT,sysplex=sysplexa
objectclass=racfConnect
racfuserid=NEWUSER
racfgroupid=GRP222
racfconnectgroupauthority=USE
racfconnectowner=racfid=ADMIN1,profile=USER,sysplex=sysplexa
racfconnectauthdate=00.224
racfconnectcount=00
racfconnectgroupuacc=NONE
racfconnectlastconnect=UNKNOWN
racfconnectattributes=NONE
racfconnectrevokedate=NONE
racfconnectresumedate=NONE
```

To see all the groups that newuser is connected to, either of the following searches can be performed:

```
ldapsearch -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,sysplex=sysplexa -w passwd
  -b "profiletype=connect,sysplex=sysplexa" "racfuserid=newuser"
```

or

```
ldapsearch -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,sysplex=sysplexa -w passwd
  -b "profiletype=connect,sysplex=sysplexa" "(&(racfuserid=newuser)(racfgroupid=*))"
```

For both commands, the results are:

```
racfuserid=NEWUSER+racfgroupid=GROUPID,profiletype=CONNECT,sysplex=sysplexa

racfuserid=NEWUSER+racfgroupid=GRP222,profiletype=CONNECT,sysplex=sysplexa
```

Note that GROUPID was the default group to which newuser was connected when newuser was created.

Example: removing a user from a group in RACF

The following command removes `newuser` from the `grp222` group (the equivalent of the RACF **remove** command):

```
ldapdelete -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,sysplex=sysplexa
  -w passwd "racfuserid=newuser+racfgroupid=grp222,profiletype=connect,sysplex=sysplexa"
```

Example: removing a user from RACF

The following command removes the `newuser` user profile from RACF, also removing all of `newuser`'s connections to groups (the equivalent of a RACF **deluser** command):

```
ldapdelete -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,sysplex=sysplexa
  -w passwd "racfid=newuser,profiletype=user,sysplex=sysplexa"
```

Deleting attributes

If a request is made to delete the **racfAttributes** attribute and no values are provided, SDBM generates a command to delete the following values (even if the user does not currently have that value): ADSP, AUDITOR, GRPACC, OIDCARD, OPERATIONS, SPECIAL, UAUDIT. Similarly, a request to delete the **racfConnectAttributes** attribute with no values results in a command to delete the following values: ADSP, AUDITOR, GRPACC, OPERATIONS, SPECIAL. Deleting a specific value for **racfAttributes** or **racfConnectAttributes** requires that the value itself be specified on the delete operation.

For example, to remove the OPERATIONS and AUDITOR values from the **racfAttributes** values of user ID `user1` (leaving any other **racfAttributes** values the user has), you would issue an **ldapmodify** command with the following file:

```
dn: racfid=user1,profiletype=user,sysplex=sysplexa
changetype: modify
delete: racfAttributes
racfAttributes: OPERATIONS
racfAttributes: AUDITOR
```

To remove all the **racfAttributes** values listed above of user ID `user1`, you would issue an **ldapmodify** command with the following file:

```
dn: racfid=user1,profiletype=user,sysplex=sysplexa
changetype: modify
delete: racfAttributes
```

In addition, you can use the **racfAttributes** attribute to remove an entire segment from a user. For example, to remove the OMVS segment from user ID user1, you would issue an **ldapmodify** command with one of the following files:

```
dn: racfid=user1,profiletype=user,sysplex=sysplexa
changetype: modify
delete: racfAttributes
racfAttributes: OMVS
```

or

```
dn: racfid=user1,profiletype=user,sysplex=sysplexa
changetype: modify
add: racfAttributes
racfAttributes: NOOMVS
```

Following are some additional examples of deleting attributes:

- ```
  dn: racfid=user1,profiletype=user,sysplex=sysplexa
  changetype: modify
  delete: racfProgrammerName
  ```

  Returns: LDAP_UNWILLING_TO_PERFORM

  The **racfProgrammerName** attribute is one that cannot be deleted.
  ```

- ```
  dn: racfid=user1,profiletype=user,sysplex=sysplexa
  changetype: modify
  delete: racfBuilding
  racfBuilding: 001
  ```

  Returns: LDAP_UNWILLING_TO_PERFORM

  You cannot specify a value to be removed for **racfBuilding**.
- ```
  dn: racfid=user1,profiletype=user,sysplex=sysplexa
  changetype: modify
  delete: racfBuilding
  ```

  Expected result: successful removal of the attribute **racfBuilding** and LDAP_SUCCESS returned.

Chapter 17. Kerberos authentication

The z/OS LDAP server allows clients to authenticate to the server by using IBM's Network Authentication and Privacy Service which is better known as Kerberos Version 5. Kerberos is a trusted third party, private-key, network authentication system. In Kerberos, a ticket, a packet of information used by a client to prove its identity, is passed to a server in place of a user name and password. This ticket is encrypted and cannot be duplicated. After the server verifies the client ticket, it sends its own ticket to the client in order for the client to authenticate it. Once the mutual authentication process is complete, the client and server have authenticated each other.

In the z/OS LDAP server, Kerberos is used for authentication only. The Kerberos options for integrity and confidentiality are not supported. Authorization information for ACLs is gathered by the LDAP server after the authentication process has completed and is based on the Kerberos identity of the bound client. For more information about Kerberos refer to *z/OS Integrated Security Services Network Authentication Service Administration* and *z/OS Integrated Security Services Network Authentication Service Programming*.

Setting up for Kerberos

Kerberos Version 5 binds, defined in IETF RFC 2222, are performed using the Generic Security Services Application Programming Interface (GSS API) defined in IETF RFCs 2743 and 2744. From this point on the phrase "GSS API bind" is used to refer to Kerberos Version 5 binds. Before attempting to perform a Kerberos GSS API bind, be sure to:

1. Have the Network Authentication and Privacy Service (Kerberos 5) installed and configured and the service started.
2. Create a Kerberos identity for the user ID that will start the LDAP server. For example:

   ```
   ALTUSER LDAPSRV PASSWORD(password) NOEXPIRED KERB(KERBNAME(ldap_prefix/hostname))
   ```

   where *ldap_prefix* is either ″LDAP″ or ″ldap″ and *hostname* is the primary hostname for the system in DNS.
3. If the KDC (Key Distribution Center) is not located on the same machine as the LDAP server, you have to generate a keytab file for the server. To generate a keytab for the server, issue the following commands:
   a. First check the version of the server's Kerberos key (this is necessary since the version is updated every time the password is changed):

      ```
      LISTUSER LDAPSRV NORACF KERB
      ```

   b. Now the **keytab** command can be issued from the z/OS shell with the version from the **LISTUSER** command:

      ```
      keytab add LDAP/hostname -p password -v 001
      ```

      The **-k** *filename* option may also be used if you want to use your own keytab file rather than the Kerberos default keytab file. It is also important to note that when issuing Kerberos commands all passwords must be in uppercase.

   If the KDC and LDAP server are on the same system, you do not need a keytab file. If the ID which starts the LDAP server has READ access to the IRR.RUSERMAP facility class in RACF, then this can be used instead of a keytab file. Following are the RACF commands to do this:

   ```
   RDEFINE FACILITY IRR.RUSERMAP UACC(NONE)
   PERMIT IRR.RUSERMAP CLASS(FACILITY) ID(LDAPSRV) ACCESS(READ)
   SETR RACLIST(FACILITY) REFRESH
   ```
4. Enable your configuration file for Kerberos authentication.

   ```
   # Global Section
   supportKrb5 yes
   serverKrbPrinc LDAP/hostname@MYREALM.COM
   ```

```
krbLDAPAdmin ibm-kn=ldapadm@MYREALM.COM
krbKeytab none
# TDBM Section
krbIdentityMap on
# SDBM Section
krbIdentityMap on
```

Note: The ″LDAP″ portion of the **serverKrbPrinc** identity can either be ″ldap″ or LDAP″ in the configuration file and in the Kerberos segment of the RACF ID where it is defined. Check your KDC for case requirements.

5. Start your server. Your LDAP server is now configured with Kerberos support.

Updating the directory schema for Kerberos

In order to enable Kerberos GSS API Authentication, the directory schema must contain the Kerberos related schema elements which are defined in **schema.user.ldif**. For more information on schema updates see “Updating the schema” on page 179.

Table 37 lists the Kerberos object classes and attributes.

Table 37. Kerberos attributes and object classes

Attribute	Object Class	Description
krbRealmName-V2	**krbRealm-V2**	This attribute represents the Kerberos Realms of which entries in the LDAP server are members. The entry that contains this attribute also contains the **krbPrincSubtree** attribute.
krbPrincSubtree	**krbRealm-V2**	This attribute is in the same entry as the **krbRealmName-V2** attribute and it identifies the directory subtrees where entries may contain Kerberos information.
krbPrincipalName	(no object class)	The attribute is used to define the entry’s Kerberos identity. This attribute is used for identity mapping. Currently this attribute is not associated with an object class. This means that for an entry to contain this attribute you can add the object class **extensibleObject** or define and add your own object class.
krbAliasedObjectName	**krbAlias**	This attribute allows an entry to be mapped to another entry’s DN.
krbHintAliases	**krbAlias**	This attribute is used as an authorization list. If another entry’s DN is in this list and that entry specified this entry as a **krbAliasedObjectName** then the mapping is allowed.
altSecurityIdentities	**ibm-securityIdentities**	If a user is defined to a case-insensitive Kerberos server, then the Kerberos identity associated with this entry is stored as an **altSecurityIdentity** rather than a **krbPrincipalName**.
ibm-kn	(no object class)	This attribute is a pseudo-DN so that Kerberos identities can be represented as DNs for access control. Currently this attribute is not associated with an object class. This means that for an entry to contain this attribute you can add the object class **extensibleObject** or define and add your own object class.

Identity mapping

The following sections describe the mapping that is done depending on your configuration. After all the identity mapping takes place you are left with a list of DNs that are used for access control and group gathering.

Default mapping

The GSS API bind operation passes a Kerberos identity to the LDAP server which in its initial form cannot be used for access control in the server. This Kerberos identity known as <*principal*>@<*REALM*> is converted to a DN of the form ibm-kn=<*principal*>@<*REALM*>. Now this Kerberos DN can be used in access control lists.

For example, if you performed a Kerberos bind as jeff@IBM.COM you would be mapped to ibm-kn=jeff@IBM.COM and this DN is added to a list of DNs that will be used for access control throughout the server. This is known as the default mapping and is always performed when a SASL bind with a mechanism of GSS API is performed.

TDBM mapping

Another form of mapping is to map the Kerberos identity to TDBM DNs. The following algorithm is used to perform this type of identity mapping if the **krbIdentityMap** configuration option is **on** for this backend.

1. Search the entire TDBM backend for the realm entry that corresponds to the Kerberos identity. This is done by searching for objectclass = krbRealm and krbRealmName-V2 = <*REALM*>, where <*REALM*> is the realm portion of the bound Kerberos identity. If the realm is found in the directory, then all of its **krbPrincSubtree** values are gathered for use in the next part of this algorithm.
2. If **krbPrincSubtree** values exist, then each subtree will be searched for the entry or entries that contain the attribute

   krbPrincipalName = <*principal*>@<*REALM*>

   where <*principal*>@<*REALM*> is the bound Kerberos identity.
3. If an entry or entries were found in step 2 with the correct **krbPrincipalName**, their DNs are added to the DN list. If the **krbAliasedObjectName** attribute exists in the entry that was found, then more work needs to be done. The entry specified as a **krbAliasedObjectName** must allow this entry to use its DN. So the entry specified in the **krbAliasedObjectName** must have the DN of the entry in its list of **krbHintAliases**. If it does, then the **krbAliasedObjectName** value is added to the DN list.
4. The final step is to search the entire database for entries that have an object class

   objectclass = ibm-securityIdentities

   and the attribute

   altSecurityIdentities = KERBEROS:<*principal*>@<*REALM*>

   where <*principal*>@<*REALM*> is the bound Kerberos identity.

SDBM mapping

If an SDBM backend is configured and the **krbIdentityMap** configuration is **on**, then the SDBM backend tries to map the Kerberos identity to the appropriate RACF ID. If a RACF ID is found, then the SDBM DN that represents the RACF ID is added to the list of DNs.

Configuring access control

Since we now have a list of alternate DNs, access control has been changed to operate on the list of DNs rather than just a single DN. Group gathering is also performed on all of the DNs in the list. The following examples show how access control could be configured for Kerberos binds.

1. Setting up new ACLs in your directory:

   Use ibm-kn=<*principal*>@<*REALM*> for your **aclEntry** values.

   Example:

   ```
   dn: cn=Scott,o=IBM,c=US
   aclEntry: access-id:ibm-kn=jeff@IBM.COM:normal:r
   ```

If jeff@IBM.COM performed a Kerberos bind to the server, he will be mapped to ibm-kn=jeff@IBM.COM and he would get read access to normal data in the Scott entry.

2. Use existing ACLs (Method 1). This method is used for Kerberos identities that are defined to IBM KDCs or case-sensitive KDCs.
   a. Set up and add the realm entry in the database.

      Example:

      ```
      dn: krbRealmName-V2=IBM.COM,o=IBM,c=US
      objectclass: krbRealm
      krbRealmName-V2: IBM.COM
      krbPrincSubtree: o=IBM,c=US
      ```

      This example states that if a bound Kerberos identity has a realm of IBM.COM, then identity mapping is performed in the o=IBM,c=US subtree.

   b. Add the **krbPrincipalName** attribute to your entries.

      Example:

      ```
      dn: cn=Jeff,o=IBM,c=US
      objectclass: extensibleObject
      krbPrincipalName: jeff@IBM.COM
      ```

      In this example, the realm object for jeff@IBM.COM is found and the o=IBM,c=US subtree is searched for krbPrincipalName = jeff@IBM.COM. Because there is no **krbAliasedObjectName** attribute in the Jeff entry, only the DN cn=Jeff,o=IBM,c=US is added to the DN list along with the default mapping of ibm-kn=jeff@IBM.COM.

      Therefore, if cn=Jeff,o=IBM,c=US was already defined in another entry's **aclEntry**, then jeff@IBM.COM will still have that access to the entry. For example:

      ```
      dn: cn=Ken,o=IBM,c=US
      aclEntry: access-id:cn=Jeff,o=IBM,c=US:normal:w
      ```

      In this example jeff@IBM.COM will still maintain access to the cn=Ken,o=IBM,c=US entry since TDBM mapping was performed.

   c. The **krbAliasedObjectName** attribute can also be used for identity mapping.

      Example:

      ```
      dn: cn=Jeff,o=IBM,c=US
      objectclass: extensibleObject
      objectClass: krbAlias
      krbPrincipalName: jeff@IBM.COM
      krbAliasedObjectName: cn=Tim,o=IBM,c=US
      ```

      In this example, the realm object for jeff@IBM.COM is found and the o=IBM,c=US subtree is searched for krbPrincipalName = jeff@IBM.COM. The search results in cn=Jeff,o=IBM,c=US being added to the DN list. Because there is a **krbAliasedObjectName** attribute in the Jeff entry, we need to look at the Tim entry before we add cn=Tim,o=IBM,c=US to the DN list. In order to use Tim's DN for access control he must authorize Jeff to do so. Tim's entry must look like the following:

      ```
      dn: cn=Tim,o=IBM,c=US
      objectclass: krbAlias
      krbHintAliases: cn=Jeff,o=IBM,c=US
      ```

      Since Tim has Jeff listed as a **krbHintAliases**, the value of **krbAliasedObjectName** cn=Tim,o=IBM,c=Us can be added to the DN list. If the Tim entry did not contain the **krbHintAliases** with Jeff as its value, then Tim's DN would not be added to the DN list.

      Therefore, if cn=Tim,o=IBM,c=US was already defined in another entry's **aclEntry** then jeff@IBM.COM will still have that access to the entry. For example:

      ```
      dn: cn=Kim,o=IBM,c=US
      aclEntry: access-id:cn=Tim,o=IBM,c=US:normal:w
      ```

In this example, jeff@IBM.COM still maintains write access to the Kim entry since TDBM mapping was performed and Jeff was aliased to Tim.

3. Use existing ACLs (Method 2). This method should be used for case-insensitive KDCs. Set up your TDBM entries with the **altSecurityIdentities** attribute.

   Example:

   ```
   dn: cn=Jeff,o=IBM,c=US
   objectclass: ibm-securityIdentities
   altSecurityIdentity: KERBEROS:jeff@IBM.COM
   ```

   Now if jeff@IBM.COM performs a Kerberos bind he will be mapped to ibm-kn=jeff@IBM.COM as well as cn=Jeff,o=IBM,c=US.

4. Therefore, if cn=Jeff,o=IBM,c=US was already defined in another entry's **aclEntry** then jeff@IBM.COM still has that access to the entry.

   For example:

   ```
   dn: cn=Ken,o=IBM,c=US
   aclEntry: access-id:cn=Jeff,o=IBM,c=US:normal:w
   ```

   In this example jeff@IBM.COM still maintains write access to the Ken entry since TDBM mapping was performed.

Example of setting up a Kerberos directory

The following diagram shows an example of how you could set up a Kerberos directory.

Note: Due to space limitations of the diagram, the entries in the example do not contain all of the necessary information to make them valid directory entries. For example, object classes and required attributes have been left out of many of the entries.

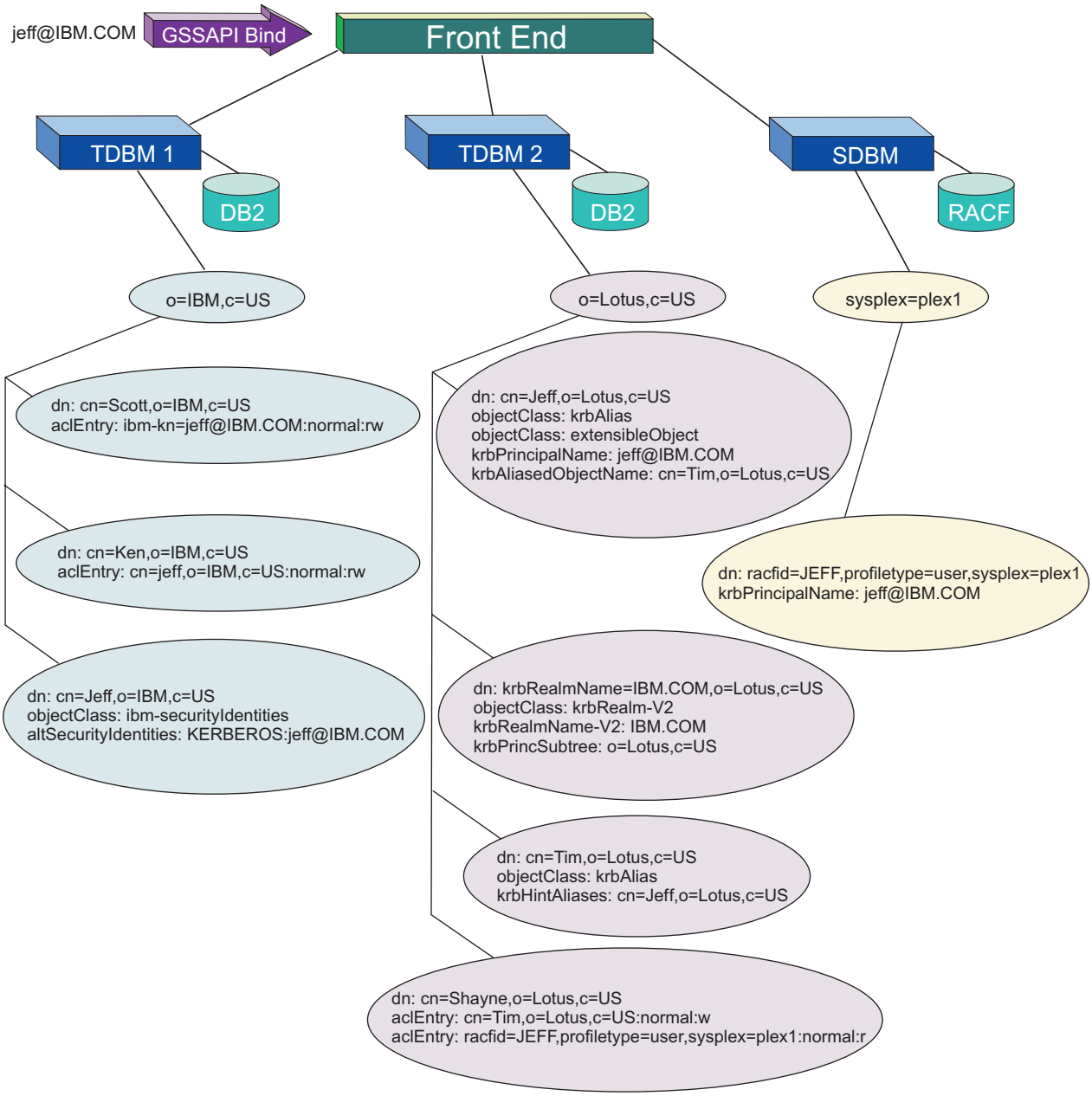


Figure 35. Kerberos directory example

Assume that Kerberos support has been enabled for this server, all backends have set **krbIdentityMap** to **on**, and the JEFF user ID has performed a **kinit** to acquire a Kerberos ticket before issuing the GSS API Kerberos bind.

The user JEFF with a Kerberos identity of `jeff@IBM.COM` is performing a Kerberos GSS API Bind to an LDAP server which has been configured with two TDBM backends and an SDBM backend.

During the bind process the Kerberos identity `jeff@IBM.COM` by default is mapped to `ibm-kn=jeff@IBM.COM` and this value is added to the list of DNs that is used for access control.

After default mapping is performed, each of the backends attempt to perform identity mapping:

1. The TDBM 1 backend first looks for the Kerberos realm object with a krbRealmName-V2 = IBM.COM and will not find one. Now the backend attempts to find the entry that contains altSecurityIdentities = KERBEROS:jeff@ibm.com. The entry with the DN cn=Jeff,o=IBM,c=US matches this criteria and the DN is added to the alternate DN list.
2. Now the server moves on to the TDBM 2 backend and tries to find the Kerberos realm object with a krbRealmName-V2 = IBM.COM. This time the realm object is found so all of the **krbPrincSubtree** values of the realm object are collected. Next, the server searches each of these subtrees (in this example, only the o=Lotus,c=US subtree) for entries that contain krbPrincipalName = jeff@IBM.COM. In this backend the entry cn=Jeff,o=Lotus,c=US is found and is added to the DN list. Next the JEFF entry is checked for the **krbAliasedObjectName** attribute. There is a **krbAliasedObjectName** specified so authorization of the alias needs to be performed. The alias is cn=Tim,o=Lotus,c=US so the Tim entry must be checked for the attribute **krbHintAliases** with a value of cn=Jeff,o=Lotus,c=US. This value does exist so the DN cn=Tim,o=Lotus,c=US is added to the access control DN list.

   Note: If the value cn=Jeff,o=Lotus,c=US did not exist in Tim's **krbHintAliases**, then Tim did not want you to alias him, so the DN cn=Tim,o=Lotus,c=US would not have been added to the DN list.
3. Finally, the server gets to the SDBM backend and invokes a RACF API that attempts to map the Kerberos identity jeff@IBM.COM to its associated RACF ID. In this example, the API returns the JEFF user ID and the DN racfid=JEFF,profiletype=user,sysplex=plex1 is constructed and added to the list of access control DNs.

At this point, the bind has completed and the list of DNs that is used for access control is as follows:

```
ibm-kn=jeff@IBM.COM
cn=Jeff,o=IBM,c=Us
cn=Jeff,o=Lotus,c=Us
cn=Tim,o=Lotus,c=US
racfid=JEFF,profiletype=user,sysplex=plex1
```

Group gathering can now be performed on the entire list of DNs.

Now that jeff@IBM.COM is bound to the server and his list of alternate DNs has been generated, he now has authority to perform other operations:

- Because jeff@IBM.COM was mapped to ibm-kn=jeff@IBM.COM he has read and write permission to normal data in the cn=Scott,o=IBM,c=US entry.
- jeff@IBM.COM also has read and write permission to the normal data in the cn=Ken,o=IBM,c=US entry due to his identity also being mapped to cn=Jeff,o=IBM,c=US.
- Modify operations would be permitted on the cn=Shayne,o=IBM,c=US entry since jeff@IBM.COM was also mapped to cn=Tim,o=Lotus,c=US and Tim has write access to Shayne.
- Read access is also permitted on the cn=Shayne,o=IBM,c=US entry because jeff@IBM.COM was mapped to the SDBM DN racfid=JEFF,profiletype=user,sysplex=plex1 who has read permission to the cn=Shayne,o=IBM,c=US entry.

You can see from this example that your access control is based on the combination of all the mapped DN's access control permissions.

Kerberos operating environments

Because Kerberos Version 5 is interoperable with other Kerberos 5 implementations, there are a variety of different operating environments that can exist.

- Another KDC other than the z/OS KDC can be used to store Kerberos principals. Users get a ticket from the other KDC rather than the z/OS KDC. The LDAP server could also be registered to this other KDC. However a trusted realm between the z/OS KDC and the external KDC must be established.
- Another KDC can be used along with the z/OS KDC where both will store Kerberos identities. In this scenario a trusted realm needs to be configured between the two realms.

- z/OS user IDs can be set up to contain an external KDC's Kerberos identities so when SDBM identity mapping is performed you can still be mapped to a RACF ID if you are strictly using the external or foreign KDC for Kerberos identities. This is done by setting up a **KERBLINK** and a trusted realm. The following **KERBLINK** example adds the foreign Kerberos identity `jeff@KRB2000.IBM.COM` to the RACF user JEFF:

  ```
  RDEFINE KERBLINK /.../KRB2000.IBM.COM/jeff APPLDATA('JEFF')
  ```

  For information about how to set up trusted realms and **KERBLINK**, refer to *z/OS Integrated Security Services Network Authentication Service Administration*.

Chapter 18. Native authentication

LDAP has the ability to authenticate to the Security Server through TDBM by supplying a Security Server password on a simple bind to a TDBM backend. Authorization information is still gathered by the LDAP server based on the DN that performed the bind operation. The LDAP entry that contains the bind DN should contain either the **ibm-nativeId** attribute or **uid** attribute to specify the ID that is associated with this entry. Note that the SDBM backend does not have to be configured. The ID and password are passed to the Security Server and the verification of the password is performed by the Security Server. Another feature of native authentication is the ability to change your Security Server's password by issuing an LDAP modify command.

Note: The use of RACF passtickets is supported by the z/OS LDAP server. It is recommended that the LDAP server be run as a started task if RACF passticket support will be used. The job name associated with the LDAP server started task should be used as the application name when generating RACF passtickets. Refer to *z/OS Security Server RACF Macros and Interfaces* for more information about RACF passtickets.

Initializing native authentication

To enable native authentication, perform the following steps:

1. Install and configure RACF or another Security Server.
2. Configure an LDAP server to run TDBM and start the server. Specify the native authentication options in your configuration file. For example:

```
TDBM Section
  useNativeAuth SELECTED
  nativeAuthSubtree o=IBM,c=US
  nativeAuthSubtree o=Lotus,c=US
  nativeUpdateAllowed YES
```

3. Load the native authentication related schema elements into the TDBM backend.
4. Be sure that the entries that are to perform native authentication contain either the **ibm-nativeId** attribute or a single-valued **uid** attribute with the appropriate Security Server ID as its value. It is important to note that a multi-valued **uid** without an **ibm-nativeId** causes the bind to fail because the LDAP server does not know which ID to use.

Updating the schema for native authentication

In order to enable native authentication, the directory schema must contain the native authentication related schema elements which are defined in **schema.IBM.ldif**. For more information on schema updates see "Updating the schema" on page 179.

Following is the native authentication attribute type:

ibm-nativeId
: Allows you to specify the ID that is to be associated with this entry.

Following is the native authentication object class:

ibm-nativeAuthentication
: Allows you to specify the **ibm-nativeId** attribute in entries.

Defining participation in native authentication

There are many different configuration options for native authentication which are discussed in this section.

The main configuration option, **useNativeAuth**, can be set to **selected**, **all**, or **off**. If you want all entries in a certain subtree to participate in native authentication then you would choose **all** for this option. However, if you would like specific entries in the specific subtrees to be subject to native authentication, then choose **selected** for the **useNativeAuth** option. When **selected** is used, only entries with the **ibm-nativeId** attribute will be subject to native authentication.

In order for an entry to bind natively or perform a native password modify, that entry must contain a mapping to the Security Server identity that is associated with the user. This can be accomplished by using either the **ibm-nativeId** attribute or the **uid** attribute that is defined in **schema.user.ldif**. If your directory entries already contain a single-valued **uid** attribute (which holds the Security Server user ID), then these entries are already configured for native authentication if you plan on using the **useNativeAuth all** option. If you do not plan on using **uid**s for mapping, then you can specify the **ibm-nativeId** attribute for your Security Server ID associations and this attribute is used with **selected** or **all** specified for the **useNativeAuth** option. If both the **ibm-nativeId** and **uid** attributes exist in an entry, the **ibm-nativeId** value is used. The userid specified by either the **uid** or **ibm-nativeId** attributes must contain a valid OMVS segment in the Security Server.

If you use the **useNativeAuth** option, also specify the **nativeUpdateAllowed** option to enable native password changes in the Security Server to occur through the TDBM backend.

Next, consider what portions of your directory should have the ability to participate in native authentication. If the entire directory should participate, then set the **nativeAuthSubtree** configuration option to **all**. If there are different subtrees in your directory which contain entries that need to bind natively or perform native password modifications, then you need to list all the subtrees with the **nativeAuthSubtree** configuration option.

Note: If the DN that is listed in the **nativeAuthSubtree** option contains a space character in it, then the entire DN must be enclosed in quotes in the configuration file.

As mentioned above, there are two LDAP operations affected: bind and password modify. There is a set of criteria that is used to determine if an entry actually participates in native authentication. This criteria changes depending on the configuration options that have been selected. The following table outlines all the possible operating modes for native authentication.

Table 38. Operating modes for native authentication

Operation	useNativeAuth	nativeUpdate Allowed	ibm-nativeId	uid	Behavior
Bind	selected	any value	User1		Entry is configured correctly and native authentication is attempted.
Bind	selected	any value		User1	Entry is not correctly configured for native authentication so an LDAP simple bind is attempted. The **uid** attribute is not used when **useNativeAuth** is **selected**.
Bind	selected	any value			Entry has not been configured for native authentication so an LDAP simple bind is attempted.
Bind	all	any value	User1	User2	The **ibm-nativeId** attribute is used to attempt native authentication.
Bind	all	any value		User1	Entry is configured correctly and native authentication is attempted.

Table 38. Operating modes for native authentication (continued)

Operation	useNativeAuth	nativeUpdate Allowed	ibm-nativeId	uid	Behavior
Bind	all	any value			For ease of implementation, a LDAP simple bind is attempted, even though you have specified that all entries should use native authentication. Eventually this entry should be configured correctly.
Modify/ Replace (password)	selected	Yes	User1		Operation is not allowed because the entry is configured for native authentication. A modify/delete followed by a modify/add must be performed.
Modify/ Replace (password)	selected	Yes			Entry is not configured for native authentication so a regular LDAP password replace is attempted.
Modify/ Replace (password)	all	Yes			Operation is not allowed. modify/delete followed by a modify/add must be performed.
Modify/ Delete (password)	selected	Yes	User1		Entry is configured for native authentication so the value specified is used to change USER1 Security Server password if a modify/add follows this operation. If a modify/add does not follow, then the operation fails.
Modify/ Delete (password)	selected	Yes			Entry is not configured for native authentication so a regular LDAP modify/delete is attempted.
Modify/ Delete (password)	all	Yes	User1	User2	Entry is configured for native authentication so the value specified is used to change USER1 Security Server password if a modify/add follows this operation. If a modify/add does not follow, then the operation fails.
Modify/ Delete (password)	all	Yes		User1	Entry is configured for native authentication so the value specified is used to change USER1 Security Server password if a modify/add follows this operation. If a modify/add does not follow, then the operation fails.
Modify/ Delete (password)	all	Yes			A regular LDAP modify/delete will be allowed in this case to allow for old LDAP passwords stored in TDBM to be removed.

Table 38. Operating modes for native authentication (continued)

Operation	useNativeAuth	nativeUpdate Allowed	ibm-nativeId	uid	Behavior
Modify/ Add (password)	selected	Yes	User1		If a password modify/delete was previously performed, then a Security Server password change for USER1 is attempted. If the modify/delete had not been performed, then the operation fails. Also, if the Security Server ID is not defined, the operation fails.
Modify/ Add (password)	selected	Yes			Entry is not configured for native authentication so a regular LDAP modify/add is attempted.
Modify/ Add	all	Yes	User1	User2	If a password modify/delete was previously performed then a Security Server password change for USER1 is attempted. If the modify/delete had not been performed, then the operation fails. Also, if the Security Server ID is not defined, the operation fails.
Modify/ Add	all	Yes		User1	If a password modify/delete was previously performed, then a Security Server password change for USER1 is attempted. If the modify/delete had not been performed, then the operation fails. Also, if the Security Server ID is not defined, the operation fails.
Modify/ Add	all	Yes			Operation fails because the entry is not correctly configured for native authentication.
Add (entry with password)	selected		User1		Entry is configured for native authentication so adding an entry with a password is not allowed.
Add (entry with password)	selected			User1	Entry is not configured for native authentication so the operation is attempted.
Add (entry with password)	selected				Entry is not configured for native authentication so the add operation is attempted.
Add (entry with password)	all		User1	User2	Operation fails. Native entries do not need LDAP passwords.
Add (entry with password)	all			User1	Operation fails. Native entries do not need LDAP passwords.
Add (entry with password)	all				Operation fails. Native entries do not need LDAP passwords.
Notes: This table assumes that the entry is located within native authentication subtrees. If a Security Server ID has been specified for a native entry but has not yet been defined in Security Server, the LDAP server will attempt an LDAP simple bind. If a native entry is configured with a multi-valued **uid** and no **ibm-nativeId**, then the operation fails because the LDAP server does not know which value to use as the user ID.					

Updating native passwords

Performing a native password modify is as simple as issuing an **ldapmodify** command to perform a delete followed by an add of the **userpassword** attribute. Specify the current password on the delete statement followed by the new password on the add statement. The delete must occur before the add for native password modify. For example, if the file `pw.mod` contains:

```
cn=You,o=IBM,c=US
-userpassword=oldpassword
+userpassword=newpassword
```

then the following command modifies the native password (assuming the bind DN has the authority to do this):

```
ldapmodify ... -D cn=You,o=IBM,c=US -w oldpassword -f pw.mod
```

The following *errno* values returned by **__passwd()** will have an LDAP reason code defined for them:

Table 39. The errno values returned by __passwd()

errno **value**	**Reason**	**Text**
EMVSERR	R004107	The password function failed; not loaded from a program controlled library.
EMVSSAF2ERR	R004108	TDBM backend password API resulted in an internal error.
EMVSEXPIRE	R004109	The password has expired.
EMVSSAFEXTRERR	R004110	The userid has been revoked.
EACCES	R004111	The password is not correct.
EINVAL	R004112	A bind argument is not valid.
ESRCH	R004118	Entry native user ID (**ibm-nativeId**,**uid**) is not defined to the Security Server.
EMVSPASSWORD	R004128	Native authentication password change failed. The new password is not valid or does not meet requirements.

If the **ldapmodify** command above fails with LDAP return code **LDAP_INVALID_CREDENTIALS** and LDAP reason code:

```
R004109 The password has expired.
```

then it is possible to change the RACF password of a TDBM entry participating in native authentication by doing an LDAP simple bind. The simple bind can occur as part of an LDAP function such as search, add, or modify. The password change is provided in the password portion of the LDAP simple bind. The password must be in the following format:

```
password/newpassword
```

The forward slash (/) is used as the indication of a password change during the LDAP simple bind. Password changes made using the LDAP simple bind to a TDBM entry participating in native authentication are subject to the system password rules. A password change will fail with LDAP return code **LDAP_INVALID_CREDENTIALS** and LDAP reason code of:

```
R004128 Native authentication password change failed: The new password is not valid, or does not
meet requirements.
```

if the new password does not pass the rules established on the system.

Note that once the bind succeeds, the password is changed even if the LDAP function eventually fails.

Assuming TDBM entry `cn=User1,ou=END,o=IBM,c=US` is participating in native authentication, the following command changes the RACF password for user `USER1` from `abc` to `def`:

```
ldapsearch -h ldaphost -p ldapport -D "cn=User1,ou=END,o=IBM,c=US" -w abc/def -b "ou=END,o=IBM,c=US"\
"objectclass=*"
```

Note: LDAP ACLs must be set properly to allow update of the **userpassword** attribute for the password modification to complete successfully. The distinguished name provided on the **-D** parameter of the **ldapmodify** command must have authority to update the **userpassword** attribute. To allow each individual user to update their own password, an LDAP ACL should be established to permit them to write **userpassword** attribute values.

It is possible to use the special `cn=this` identity to establish the LDAP ACL.

Run the following **ldapmodify** command to establish the LDAP ACL:

```
ldapmodify -D adminDN -w adminPW -f /tmp/aclmod.ldif
```

where the file /tmp/aclmod.ldif looks like:

```
dn: o=Your Company
changetype: modify
add: x
aclEntry: access-id:cn=this:critical:rwsc
aclPropagate: TRUE
```

You should substitute the root of your directory tree for the `dn: o=Your Company` line in the LDIF file. This will allow each user defined for native authentication to update their own RACF password via LDAP.

Example of setting up native authentication

The following diagram shows an example of how you could set up native authentication.

Note: Due to space limitations of the diagram, the entries in the example do not contain all of the necessary information to make them valid directory entries. For example, object classes and required attributes have been left out of many of the entries.

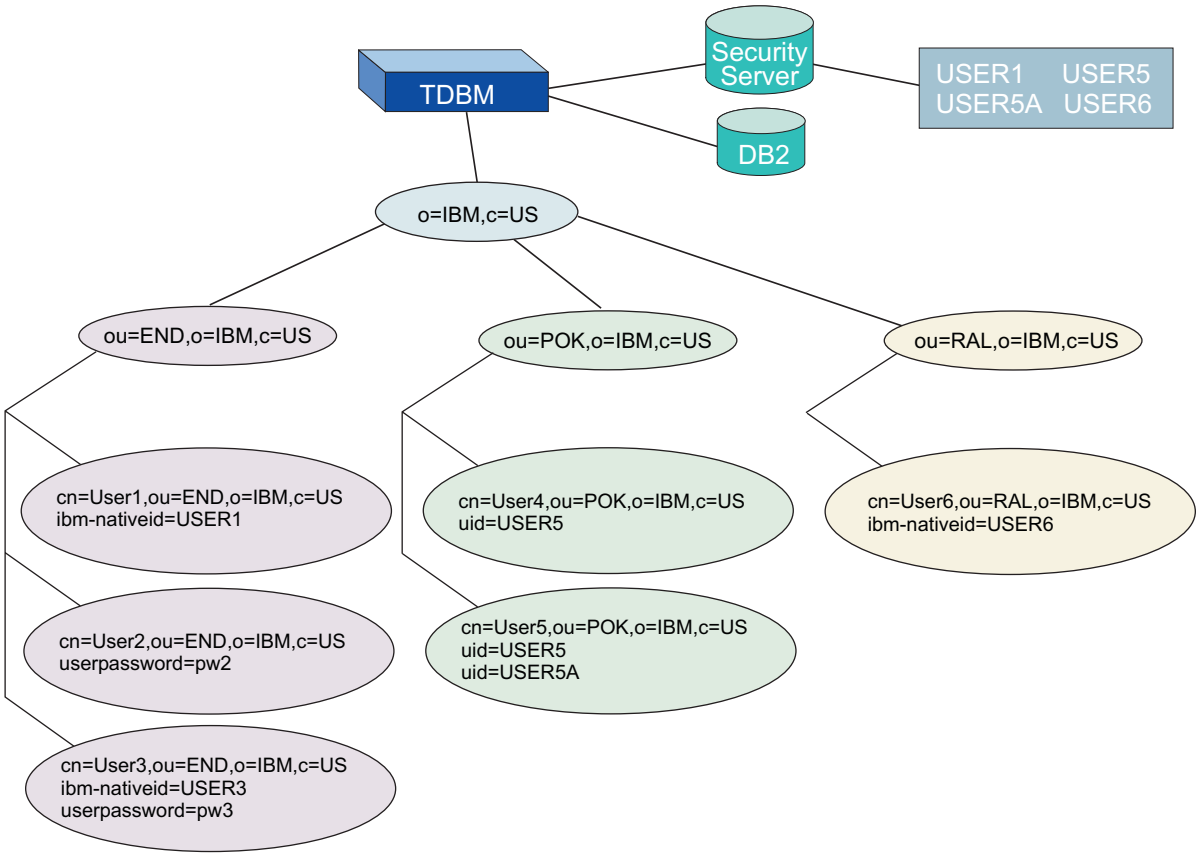


Figure 36. Native authentication example

Example 1

- Assuming these settings:
  useNativeAuth selected
  nativeUpdateAllowed yes
  nativeAuthSubtree ou=END,o=IBM,c=US
  nativeAuthSubtree ou=POK,o=IBM,c=US

  the following table indicates the results of operations involving each user entry:

Table 40. Behavior of native authentication in example 1

LDAP entry	Operation	Behavior
cn=User1,ou=END,o=IBM,c=US	Bind	Can bind natively because the entry contains a valid **ibm-nativeId**.
	Native password change	Can change this native password because the entry contains a valid **ibm-nativeId**.
	Modify/replace (**userpassword**)	Cannot perform a modify/replace of **userpassword** because the entry is subject to native authentication and password replace is not allowed.
cn=User2,ou=END,o=IBM,c=US	All	Entry is not configured for native authentication so all operations are regular LDAP operations.
cn=User3,ou=END,o=IBM,c=US	Bind	Attempts native authentication but because the Security Server ID is not defined, a regular LDAP bind is performed.

Table 40. Behavior of native authentication in example 1 (continued)

LDAP entry	Operation	Behavior
	Native password change	Native password change is attempted but eventually fails because the Security Server ID USER3 is not defined.
	Modify/replace (**userpassword**)	An attempt to modify/replace the **userpassword** attribute will fail because the entry is configured for native authentication.
cn=User4,ou=POK,o=IBM,c=US	All	Performs regular LDAP operations because the entry does not contain the **ibm-nativeId** attribute.
cn=User5,ou=POK,o=IBM,c=US	All	Performs regular LDAP operations because the entry does not contain the **ibm-nativeId** attribute.
cn=User6,ou=RAL,o=IBM,c=US	All	Performs regular LDAP operations because the entry does not exist in a native subtree.

Example 2

- Now, assuming these settings:
  useNativeAuth all
  nativeUpdateAllowed yes
  nativeAuthSubtree ou=END,o=IBM,c=US
  nativeAuthSubtree ou=POK,o=IBM,c=US

  the following table indicates the results of operations involving each user entry:

Table 41. Behavior of native authentication in example 2

LDAP Entry	Operation	Behavior
cn=User1,ou=END,o=IBM,c=US	Bind	Can bind natively because the entry contains a valid **ibm-nativeId**.
	Native password change	Can change this native password because the entry contains a valid **ibm-nativeId**.
	Modify/replace (**userpassword**)	Cannot perform a modify/replace of **userpassword** because the entry is subject to native authentication and password replace is not allowed.
cn=User2,ou=END,o=IBM,c=US	Bind	Because there are no native attributes in this entry, a regular LDAP bind is attempted.
	Native password change	An attempt to change the **userpassword** attribute fails because the entry is not correctly configured for native authentication. The entry should contain either the **ibm-nativeId** attribute or **uid** attribute.
	Modify/replace (**userpassword**)	Modify/replace of **userpassword** is not allowed because **useNativeAuth** is **all**.
cn=User3,ou=END,o=IBM,c=US	Bind	Attempts native authentication but because the Security Server ID is not defined, a regular LDAP bind is performed.
	Native password change	Native password changes are attempted but eventually fail because the Security Server ID is not defined.
	Modify/replace (**userpassword**)	An attempt to modify/replace the **userpassword** attribute will fail because the entry is configured for native authentication.

Table 41. Behavior of native authentication in example 2 (continued)

LDAP Entry	Operation	Behavior
cn=User4,ou=POK,o=IBM,c=US	Bind	Can bind natively because the entry contains the **uid** attribute.
	Native password change	Can change the native password because the entry contains a valid **uid**.
	Modify/replace (**userpassword**)	An attempt to modify/replace the **userpassword** attribute will fail because the entry is configured for native authentication.
cn=User5,ou=POK,o=IBM,c=US	Bind	Native bind fails because 2 **uid** values exist.
	Native password change	Change password fails because 2 **uid** values exist.
	Modify/replace (**userpassword**)	Modify/replace of **userpassword** attribute is not allowed because **useNativeAuth** is **all**.
cn=User6,ou=RAL,o=IBM,c=US	All	Performs regular LDAP operations because the entry does not exist in a native subtree.

Using native authentication with Web servers

Many Web servers provide a user ID and password challenge for authentication. These can take advantage of native authentication. The Web server must be configured to do LDAP authentication. When the challenge to do LDAP authentication is presented, the user can enter the Security Server user ID and password (from the system where the LDAP server is running). The Web server will search the LDAP directory for an entry where **uid** equals the input user ID. The Web server will use the returned DN and the inputted password to do an **ldap_simple_bind()**. When the LDAP server determines this entry is subject to native authentication, it will retrieve the **ibm-nativeId** or **uid** value and verify the password with the Security Server. Note that if **useNativeAuth** is set to **selected**, it may be necessary to place the Security Server user ID into both the **uid** and **ibm-nativeId** attributes of this entry to allow the Web server processing to work correctly with native authentication.

Chapter 19. CRAM-MD5 and DIGEST-MD5 Authentication

The z/OS LDAP server allows clients to authenticate to the server by using CRAM-MD5 (Challenge Response Authentication Mechanism - RFC 2104) and DIGEST-MD5 (RFC 2831) SASL bind mechanisms. CRAM-MD5 and DIGEST-MD5 bind mechanisms are multi-stage binds where the server sends the client a challenge and then the client sends a response back to the server to complete authentication. The client response contains a hash, which is encoded according to the specifications of the DIGEST-MD5 or CRAM-MD5 RFC, and a username. The username that is specified on the client is also known as the authentication identity, which is used to perform the authentication with the server. On the z/OS LDAP server, the username is the **uid** attribute value of an entry that is targeted for authentication. When the server gets the response from the client, the response is parsed and the server calculates its own hash with the password found for the **uid** attribute value in the backend. If the server hash is equal to the client hash, then the client and the server are authenticated.

DIGEST-MD5 and CRAM-MD5 SASL bind mechanisms are much better than simple binds since they do not pass the credentials in the clear, which are more liable to interception by snoopers. Also, CRAM-MD5 and DIGEST-MD5 bind mechanisms on the z/OS LDAP server do not require any additional products to be installed or configured in order to have this added functionality.

DIGEST-MD5 authentication is much stronger than CRAM-MD5 authentication because it prevents chosen plaintext password attacks. During a DIGEST-MD5 authentication, there is additional information that is passed between the server and the client during the challenges and responses than compared to the CRAM-MD5 algorithm. Thus, it is more difficult to use the DIGEST-MD5 hashing algorithm to decipher the password for the authentication identity than the CRAM-MD5 hashing algorithm.

DIGEST-MD5 restrictions on the z/OS LDAP server:

1. The options for integrity and encryption protection on DIGEST-MD5 binds are not supported.
2. The unspecified userid form of the authorization identity is not supported; however, the DN version is supported on the z/OS LDAP client and server.
3. An authorization DN is used to obtain the authority of another LDAP user after successfully authenticating based upon the authentication identity. The z/OS LDAP server does not support specifying an authorization DN that is different than the authentication identity's DN.
4. Subsequent authentication is not supported.

Considerations for setting up a TDBM backend for CRAM-MD5 and DIGEST-MD5 Authentication

The following are considerations for setting up a TDBM backend for CRAM-MD5 and DIGEST-MD5 authentication:

1. In order to use CRAM-MD5 and DIGEST-MD5 authentication on the z/OS LDAP server, the entries that you bind with must contain a **uid** attribute value. The **uid** attribute value is used by the z/OS LDAP server as the authentication identity or username when attempting a CRAM-MD5 or DIGEST-MD5 authentication bind. The **uid** attribute is found in the schema file: **/usr/lpp/ldap/etc/schema.user.ldif**. It is strongly suggested that the **uid** attribute values be unique across every TDBM backend that is configured on the LDAP server. When attempting a CRAM-MD5 or DIGEST-MD5 bind, the username that is specified on the z/OS LDAP operation utilities must correspond to the **uid** attribute value of the entry that you want to authenticate as. A search of every TDBM backend that is configured on the LDAP server will be done to look for the **uid** attribute or username value that was entered on the client. If this search finds more than one DN entry that has the same **uid** value, then the bind will not be successful because it is not known which entry is targeted for authentication. However, this problem can be avoided by specifying an authorization DN. If the authentication identity or **uid** attribute value is present in the authorization DN's entry, then the bind will be successful assuming that the password is correct; otherwise it will fail.

2. In order for CRAM-MD5 and DIGEST-MD5 binds to work properly, the **userpassword** attribute for the entry must be in either clear text (not recommended) or in the DES 2-way encryption algorithm (recommended). See Chapter 7, “Preparing the backends, SSL/TLS, and password encryption,” on page 41.
3. CRAM-MD5 and DIGEST-MD5 binds are not supported with entries that are participating in Native Authentication.
4. CRAM-MD5 and DIGEST-MD5 binds are not supported to the SDBM backend.

CRAM-MD5 and DIGEST-MD5 configuration parameter

The **digestRealm** optional configuration parameter allows for the specification of a realm name to be used to help create the CRAM-MD5 and DIGEST-MD5 hashes. The value of this parameter gets passed on the initial challenge from the server to the client once it has been decided that a CRAM-MD5 or DIGEST-MD5 bind is desired. See the **digestRealm** parameter in the “Configuration file options” on page 56. This parameter defaults to the fully qualified hostname of the system where the LDAP server is running assuming that a DNS (Domain Name Server) is available or it defaults to the name of the host processor.

Example of setting up for CRAM-MD5 and DIGEST-MD5

The following diagram shows an example of how you could set up your entries in your TDBM backend(s).

Note: Due to space limitations of the diagram, the entries in the example do not contain all of the necessary information to make them valid directory entries. For example, object classes and required attributes have been left out of many of the entries.

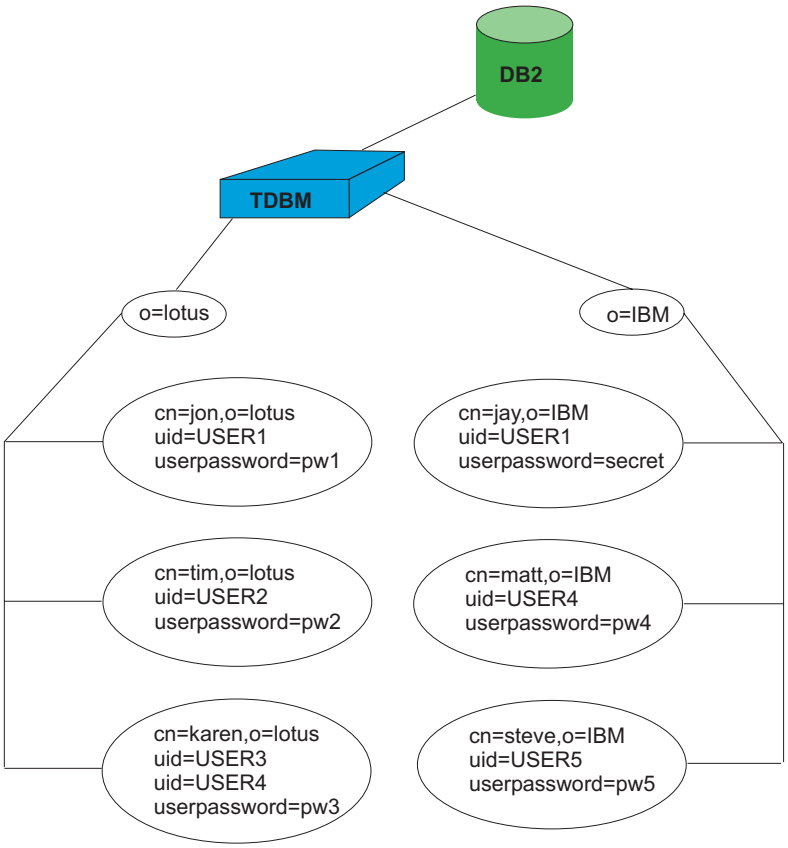


Figure 37. CRAM-MD5 and DIGEST-MD5 authentication example

The following table outlines what would happen if you attempt to do a CRAM-MD5 or DIGEST-MD5 bind from a client. The authentication identity or username refers to the **-U** option on the z/OS LDAP operation

utilities, while the authorization DN is the **-D** option on the z/OS LDAP operation utilities. See *z/OS Integrated Security Services LDAP Client Programming* for more details on the LDAP operation utilities. This table assumes that native authentication is not turned on under the subtrees: `o=lotus` and `o=IBM`.

Table 42. Behavior of CRAM-MD5 and DIGEST-MD5 authentication in example

Authentication Identity	**Authorization DN**	**Password**	**Behavior**
USER2		pw2	Bind is successful to `cn=tim,o=lotus`
USER2	`cn=tim,o=lotus`	pw2	Bind is successful to `cn=tim,o=lotus`
USER2	`cn=jon,o=lotus`	pw2	Bind is not successful because the authorization DN (`cn=jon,o=lotus`) does not equal the authentication DN (`cn=tim,o=lotus`)
USER1		pw1	Bind is not successful, because there are multiple entries with the same username/**uid** value: `cn=jon,o=lotus` and `cn=jay,o=IBM`
USER1	`cn=jay,o=IBM`	secret	Bind is successful to `cn=jay,o=IBM` because the authentication DN (`cn=jay,o=IBM`) equals the authorization DN (`cn=jay,o=IBM`)
USER1	`cn=jon,o=lotus`	pw1	Bind is successful to `cn=jon,o=lotus` because the authentication DN (`cn=jon,o=lotus`) equals the authorization DN (`cn=jon,o=lotus`)
USER3		pw3	Bind is successful to `cn=karen,o=lotus`
USER4	`cn=karen,o=lotus`	pw3	Bind is successful to `cn=karen,o=lotus`
USER4	`cn=matt,o=IBM`	pw4	Bind is successful to `cn=matt,o=IBM`
USER3	`cn=karen,o=lotus`	bad	Bind is not successful to authentication DN (`cn=karen,o=lotus`) and authorization DN (`cn=karen,o=lotus`) because the password is incorrect.
USER5	`cn=nothere,o=lotus`	pw5	Bind is not successful because the authentication DN (`cn=steve,o=IBM`) does not equal the non-existent authorization DN (`cn=nothere,o=lotus`)
BAD		pw1	Bind is not successful because a **uid** value equal to BAD was not found in any of the entries in the TDBM backend.

Chapter 20. Using extended operations to access Policy Director data

The extended operations (EXOP) backend supports two extended operations that open a connection to the target LDAP server to access z/OS Policy Director data. The **IbmLDAPProxyControl** determines the target LDAP server. To set the target LDAP server when using z/OS Policy Director, use the RACF PROXY segment. See *z/OS Security Server RACF Security Administrator's Guide* for more information.

The LDAP extended operations are **GetDnForUserid** and **GetPrivileges**. These extended operations are generated when an application on z/OS calls the AZN APIs (refer to *Policy Director Authorization Services for z/OS and OS/390 Customization and Use* for more information on using the AZN APIs). When the EXOP backend receives a request for either of these two operations, it uses the required **IbmLDAPProxyControl** to open an LDAP connection to a target LDAP server that has been set up to store Policy Director data. Then, depending on the request, the EXOP backend issues LDAP requests to the target server to retrieve the appropriate data.

GetDnForUserid extended operation

For the **GetDnForUserid** extended operation, the EXOP backend retrieves all of a user ID's distinguished names (DNs) stored in the target LDAP server. The client can filter the DNs returned by the EXOP backend by specifying a search base and object class names. The sequence of events for this extended operation is:

- If the client does not specify a search base, the EXOP backend searches for the DN of all entries in all of the target server's naming contexts that contain an **ibm-nativeId** attribute set to the specified user ID and whose set of object classes include all of the optional specified object classes. If there are no naming contexts, no results will be returned.
- If the EXOP backend does not receive entries from the target LDAP server for this first set of searches, it attempts a similar set of searches, maintaining the filtering based on the optional object classes. For the second set of searches, however, instead of searching for entries with an **ibm-nativeId** attribute set to the specified user ID, it searches for entries with a **uid** attribute set to the specified user ID.

If the client does specify a search base, the EXOP backend will attempt the same sequence of searches described above, but instead of searching all of the target LDAP server's naming contexts, it only searches the naming context specified in the search base.

Appendix E, "Supported extended operations," on page 465 summarizes some different error scenarios for this extended operation and the EXOP backend's response to such scenarios.

GetPrivileges extended operation

For the **GetPrivileges** extended operation, the EXOP backend retrieves all of a subject's Policy Director data. This subject is specified by its DN. The client can specify an optional domain name if the subject does not exist in the domain named DEFAULT. Refer to "GetPrivileges" on page 467 for an ASN.1 description of all of the data that the EXOP backend retrieves when it receives this extended operations request.

To satisfy this request, the EXOP backend performs many searches then combines all of the results prior to returning it to the client. Furthermore, some of the searches may require searches across all of the target LDAP server's naming contexts. For example, to find the groups the subject is a member of, the EXOP backend performs searches under all of the target LDAP server's naming contexts. If there are no naming contexts, no search results will be returned.

Appendix E, "Supported extended operations," on page 465 summarizes some different error scenarios for this extended operation and the EXOP backend's response to such scenarios.

These extended operations are used by z/OS Policy Director. More information about z/OS Policy Director is in *Policy Director Authorization Services for z/OS and OS/390 Customization and Use*.

Chapter 21. Static, dynamic, and nested groups

The z/OS LDAP server supports group definitions. These group definitions allow for a collection of names to be easily associated for access control checking or in application-specific uses such as a mailing list. See Chapter 22, “Using access control,” on page 269 for additional information on access control checking.

Starting with the z/OS V1R6 LDAP server, support has been added for additional object classes for static, dynamic, and nested group entries (previously only one static group object class was supported during group gathering).

It is also now possible to query static, dynamic, and nested group memberships with the use of the **ibm-allMembers** and **ibm-allGroups** operational attributes. For a given group entry, the **ibm-allMembers** attribute will enumerate all of the members that belong in that group. For a given user entry, the **ibm-allGroups** attribute will determine the groups that the user has membership in.

Static groups

A static group is defined as a group where the members are defined individually. The **accessGroup, accessRole, groupOfNames,** and **ibm-staticGroup** object classes each use a multi-valued attribute called **member** to define a list of DNs that belong to the static group. The **groupOfUniqueNames** object class uses a multi-valued attribute called **uniqueMember** to define a list of DNs that belong to the static group. These attributes and object classes are shipped in **schema.user.ldif**. A typical static group entry is as follows:

```
dn: cn=ldap_team_static,o=endicott
objectclass: groupOfNames
cn: ldap_team
member: cn=jon,o=endicott
member: cn=ken,o=endicott
member: cn=jay,o=endicott
```

Dynamic groups

A dynamic group is defined as a group in which membership is determined using one or more LDAP search expressions. Each time a dynamic group is used by the LDAP server, a user’s membership in the group is decided by determining if the user entry matches any of the search expressions. The **ibm-dynamicGroup** and **groupOfURLs** object classes each use the multi-valued attribute called **memberURL** to define the LDAP search expression. These object classes and attribute are shipped in **schema.user.ldif**.

Dynamic groups allow the group administrator to define membership in terms of attributes and allow the directory itself to determine who is or is not a member of the group. For example, members do not need to be manually added or deleted when a person moves to a different project or location.

The following simplified LDAP URL syntax must be used as the value of **memberURL** attribute to specify the dynamic group search expression.

ldap:///*baseDN*[**??**[*searchScope*][**?***searchFilter*]]

where

baseDN
: Specifies the DN of the entry from which the search begins in the directory. It can be the suffix or root of the directory. This parameter is required.

searchScope
Specifies the extent of the search. The default scope is **base**.

base Returns information only about the *baseDN* specified in the URL.

one Returns information about entries one level below the *baseDN* specified in the URL. It does not include the *baseDN*.

sub Returns information about entries at all levels below and including the *baseDN*.

searchFilter
Is the filter that you want applied to the entries within the scope of the search. See **ldapsearch** in *z/OS Integrated Security Services LDAP Client Programming* for additional information on LDAP search filters. The default is ″**objectclass=***″.

Note: As the format above suggests, the host name must not be present in the syntax. The remaining parameters are just like the normal LDAP URL syntax, defined in RFC 2255 (except there is no support for extensions in the URL). Each parameter field must be separated by a ?, even if no parameter is specified. Normally, a list of attributes to return would have been included between the *baseDN* and *searchScope*. If there are any errors in the format of the **memberURL** attribute, it will not be used for dynamic group membership gathering or for **ibm-allGroups** or **ibm-allMembers** search and comparison operations. At LDAP server initialization, all valid dynamic groups in each TDBM backend are determined and a warning message is issued if an error is encountered in the format of a **memberURL** attribute value.

A typical dynamic group entry is the following:

```
dn: cn=ldap_team_dynamic,o=endicott
objectclass: groupOfURLs
cn: ldap_team_dynamic
memberURL: ldap:///o=endicott??sub?(ibm-group=ldapTeam)
```

Dynamic group search filter examples:

A single entry in which the scope defaults to base and the filter defaults to ″objectclass=*″:

```
ldap:///cn=Ricardo,ou=Endicott,o=ibm,c=us
```

The ″In Flight Systems″ team with a scope of one-level and the filter defaults to ″objectclass=*″:

```
ldap:///ou=In Flight Systems,ou=Endicott,o=ibm,c=us??one
```

A subtree search for all the support staff in Endicott:

```
ldap:///ou=Endicott,o=ibm,c=us??sub?title=*Support
```

A subtree search for all the Garcias or Nguyens whose first name begins with an ″A″:

```
ldap:///o=ibm,c=us??sub?(&(|(sn=Garcia)(sn=Nguyen))(cn=A*))
```

A search filter that includes escaped percent signs, question marks and spaces in the base DN (`o=deltawing infosystems`) and filter (`(&(percent=10%)(description=huh?))`):

```
ldap:///o=deltawing%20infosystems,c=au??sub?(&(percent=10%25)(description=huh%3f))
```

Nested groups

A nested group is defined as a group that references other group entries, which can be static, dynamic, or nested groups. The **ibm-nestedGroup** object class uses the multi-valued attribute called **ibm-memberGroup** to indicate the DNs of the groups that are referenced by the nested group. Nested groups allow LDAP administrators to construct and display group hierarchies that describe both direct and indirect group memberships. A typical nested group entry is as follows:

```
dn: cn=ldap_team_nested,o=endicott
objectclass: container
objectclass: ibm-nestedGroup
ibm-memberGroup: cn=ldap_team_static,o=endicott
ibm-memberGroup: cn=ldap_team_dynamic,o=endicott
ibm-memberGroup: cn=ldaptest_team_nested,o=endicott
```

Determining group membership

The members of a group entry are determined depending on the type of group. Note that a group can be multiple types (for instance, both dynamic and static).

1. static group: the values of the **member** attribute of the group entry if the object class of the group entry is **accessGroup, accessRole, groupOfNames,** or **ibm-staticGroup**, or the values of the **uniqueMember** attribute if the object class is **groupOfUniqueNames**.
2. dynamic group: the DN of each entry in this TDBM backend that matches the scope and search filter contained in one of the values of the **memberURL** attribute of the group entry. A dynamic search filter is ignored if the base in the search filter is not in the same TDBM backend as the dynamic group.
3. nested group: the members of each static, dynamic, or nested group for each value of the **ibm-memberGroup** attribute in the nested group entry.

Displaying group membership

Two operational attributes can be used for querying aggregate group membership. For a given group entry, the **ibm-allMembers** attribute enumerates the entire set of group membership, including static, dynamic, and nested members as described by the nested group hierarchy. For a given user entry, the **ibm-allGroups** attribute enumerates the entire set of groups to which that user has membership, including ancestor groups from nested group hierarchy. As with all operational attributes, they will only be returned if explicitly requested and can not be specified on a search filter.

The **ibm-allGroups** and **ibm-allMembers** search and comparison operations are only supported on the TDBM backend. These operations are not supported against users or groups that are present within the SDBM backend.

If the TDBM backend's DB_VERSION has not been updated to 3.0 and an **ibm-allGroups** or **ibm-allMembers** search or comparison operation is performed, only static groups with an object class of **accessGroup** are evaluated. See Chapter 10, "Migrating to a z/OS LDAP server," on page 117 for information on static, dynamic, and nested group migration.

ACL restrictions on displaying group membership

The following ACL restrictions only apply when attempting to query **ibm-allMembers** or **ibm-allGroups** operational attributes. These rules do not apply when groups are gathered from all the backends that are participating in group gathering at authentication time. The entries and attributes used to evaluate **ibm-allMembers** and **ibm-allGroups** have ACL restrictions, which have to be fully respected depending upon the authority granted to the authenticated DN. The members of a group are determined from three sources:

1. For static groups, the bound DN must have read access on the **member** or **uniqueMember** attribute if it is performing an **ibm-allMembers** or **ibm-allGroups** search operation, or compare access if performing a comparison operation. The **member** and **uniqueMember** attributes are in the **normal** access class.
2. For dynamic groups, the bound DN must have search access on all of the attributes that are present in the dynamic group filter for any of the DNs that are returned. The ACL access to the **memberURL** attribute does not matter when resolving **ibm-allMembers** or **ibm-allGroups** attributes.
3. For nested groups, there is no restriction on using the **ibm-memberGroup** attribute, but the restrictions described above apply to the groups referenced in the nested group entry. A referenced group is ignored if it is not in the same TDBM backend as the nested group.

Specifying **ibm-allMembers** or **ibm-allGroups** in a search or compare operation also requires that the bound DN have search or compare access to the **ibm-allMembers** or **ibm-allGroups** attribute. Note that the **ibm-allMembers** and **ibm-allGroups** attributes are in the **system** access class.

For more information about access control permissions, see Chapter 22, “Using access control,” on page 269.

ACL restrictions on group gathering

At authentication time, a list is created containing the static, dynamic, and nested groups of which the binding user is a member. No ACL processing is done when reading group entries for group gathering because it is not possible to know what access rights the binding user will have to any of the attributes or subtrees in the directory until all the groups have been fully determined.

Groups migration

If you would like to use the new expanded static, dynamic, and nested group definitions and you are migrating from z/OS V1R4 or an earlier LDAP server release, it is necessary for you to update your TDBM backend’s DB_VERSION to 3.0. See Chapter 10, “Migrating to a z/OS LDAP server,” on page 117 for migrating to expanded static, dynamic, and nested groups. In z/OS V1R4 or earlier LDAP releases, only static group entries with an object class of **accessGroup** and a **member** attribute equal to the binding DN were gathered at authentication time.

Group examples

Examples of adding, modifying, and deleting group entries

Adding group entries: This example creates static group entries using the **accessGroup, groupOfUniqueNames,** and **accessRole** object classes.

```
ldapadd -h 127.0.0.1 -D "cn=admin" -w xxxx -f staticGrps.ldif
```

Where staticGrps.ldif contains:

```
dn: cn=group1, o=Your Company
objectclass: accessGroup
member: cn=bob, o=Your Company
member: cn=lisa, o=Your Company
member: cn=chris, cn=bob, o=Your Company
member: cn=john, cn=bob, o=Your Company

dn: cn=group2, o=Your Company
objectclass: groupOfUniqueNames
cn: group2
uniquemember: cn=tom, o=Your Company
uniquemember: cn=dan, o=Your Company
uniquemember: cn=sam, o=Your Company
uniquemember: cn=kevin, o=Your Company

dn: cn=group3, o=Your Company
objectclass: groupOfNames
cn: group3
member: cn=david, o=Your Company
member: cn=jake, o=Your Company
member: cn=scott, o=Your Company
member: cn=eric, o=Your Company
```

This example creates a dynamic group entry that has an object class of **groupOfURLs**:

```
ldapadd -h 127.0.0.1 -D "cn=admin" -w xxxx -f dynamicGrp.ldif
```

Where dynamicGrp.ldif contains:

```
dn: cn=dynamic_team,o=Your Company
objectclass: groupOfUrls
cn: dynamic_team
memberurl: ldap:///o=YourCompany??sub?(employeeType=ldapTeam)
```

This example creates a nested group entry with an object class of **ibm-nestedGroup** that references `cn=dynamic_team,o=Your Company` and `cn=group1,o=Your Company`.

Note: The **ibm-nestedGroup** object class is an **AUXILARY** object class and also requires a **STRUCTURAL** object class.

```
ldapadd -h 127.0.0.1 -D "cn=admin" -w xxxx -f nestedGrp.ldif
```

Where nestedGrp.ldif contains:

```
dn: cn=nested_grp,o=Your Company
objectclass: ibm-nestedGroup
objectclass: person
cn: nested_grp
sn: group
ibm-memberGroup: cn=dynamic_team,o=Your Company
ibm-memberGroup: cn=group1,o=Your Company
```

Modifying group entries: In order to add a **member** to a static group, add the user's distinguished name as an additional value for the **member** or **uniqueMember** attribute. Following is an example:

```
ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modStaticGrp.ldif
```

Where modStaticGrp.ldif contains:

```
dn: cn=group1, o=Your Company
changetype: modify
add: member
member: cn=jeff, cn=tim, o=Your Company

dn: cn=group2, o=Your Company
changetype: modify
add: uniqueMember
uniqueMember: cn=joe,o=Your Company
```

In order to remove a **member** from a static group, remove the user's distinguished name from the set of **member** or **uniqueMember** attribute values in the static group entry. Following is an example:

```
ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modStaticGrp.ldif
```

Where modStaticGrp.ldif contains:

```
dn: cn=group1, o=Your Company
changetype: modify
delete: member
member: cn=jeff, cn=tim, o=Your Company

dn: cn=group2, o=Your Company
changetype: modify
delete: uniqueMember
uniqueMember: cn=joe,o=Your Company
```

In order to add a new search expression to a dynamic group, add the LDAP URL search expression as a value of the **memberURL** attribute. Following is an example:

```
ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modDynamicGrp.ldif
```

Where modDynamicGrp.ldif contains:

```
dn: cn=dynamic_team, o=Your Company
changetype: modify
add: memberURL
memberURL: ldap:///o=Your Company??sub?(employeeType=javaTeam)
```

In order to remove a search expression from a dynamic group entry, the **memberURL** attribute value containing the search expression must be removed from the group entry. Following is an example:

```
ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modDynamicGrp.ldif
```

Where modDynamicGrp.ldif contains:

```
dn: cn=dynamic_team, o=Your Company
changetype: modify
delete: memberURL
memberURL: ldap:///o=Your Company??sub?(employeeType=javaTeam)
```

In order to add a new group reference to an existing nested group entry, add the new group's DN as a value of the **ibm-memberGroup** attribute. Following is an example:

```
ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modNestedGrp.ldif
```

Where modNestedGrp.ldif contains:

```
dn: cn=nested_grp, o=Your Company
changetype: modify
add: ibm-memberGroup
ibm-memberGroup: cn=group2,o=Your Company
```

In order to remove a group reference entry from an existing nested group entry, the **ibm-memberGroup** attribute value containing the group reference DN must be deleted. Following is an example:

```
ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modNestedGrp.ldif
```

Where modNestedGrp.ldif contains:

```
dn: cn=nested_grp, o=Your Company
changetype: modify
delete: ibm-memberGroup
ibm-memberGroup: cn=group2,o=Your Company
```

Deleting group entries: In order to delete a static, dynamic, or nested group entry, delete the directory entry that represents the group. The **ldapdelete** command can be used to perform this delete operation.

This example deletes the static, dynamic, and nested group entries that were created in the above examples:

```
ldapdelete -h 127.0.0.1 -D "cn=admin" -w xxx -f deleteGrp.list
```

Where deleteGrp.list contains:

```
cn=nested_grp,o=Your Company
cn=group1,o=Your Company
cn=group2,o=Your Company
cn=group3,o=Your Company
cn=dynamic_team,o=Your Company
```

Examples of querying group membership

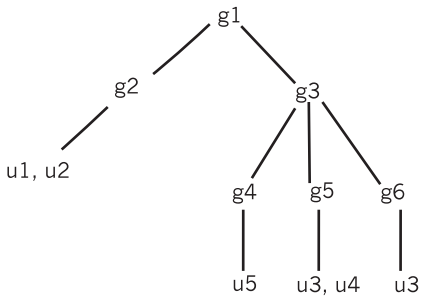


Figure 38. Group hierarchy and membership for the examples

The entries below are used in the following examples:

```
dn: o=ibm
objectclass: organization
aclEntry: group:CN=ANYBODY:normal:rsc:system:rsc
aclPropagate: TRUE
o: group

dn: cn=g1,o=ibm
objectclass: container
objectclass: ibm-nestedGroup
ibm-memberGroup: cn=g2,o=ibm
ibm-memberGroup: cn=g3,o=ibm
aclEntry: group:CN=ANYBODY:normal:rsc:system:rsc

dn: cn=g2,o=ibm
objectclass: accessGroup
member: cn=u1,o=ibm
member: cn=u2,o=ibm
aclEntry: group:CN=ANYBODY:normal:rsc:system:rsc
aclEntry: access-id:cn=u1,o=ibm:normal:rsc:system:rsc
aclEntry: access-id:cn=u2,o=ibm:normal:rsc:system:rsc:at.member:deny:rsc

dn: cn=g3,o=ibm
objectclass: container
objectclass: ibm-nestedGroup
ibm-memberGroup: cn=g4,o=ibm
ibm-memberGroup: cn=g5,o=ibm
ibm-memberGroup: cn=g6,o=ibm

dn: cn=g4,o=ibm
objectclass: accessGroup
aclEntry: group:CN=ANYBODY:normal:rsc:system:rsc
aclEntry: access- id:cn=u4,o=ibm:normal:rsc:system:rsc:at.member:deny:c
member: cn=u5,o=ibm

dn: cn=g5,o=ibm
objectclass: container
objectclass: ibm-dynamicGroup
memberURL: ldap:///o=ibm??sub?(|(cn=u3)(cn=u4))
aclEntry: group:cn=ANYBODY:normal:rsc:system:rsc
aclEntry: access-id:cn=u3,o=ibm:normal:rsc:system:rsc:at.ibm-allMembers:deny:rs:
 at.ibm-allMembers:grant:c

dn: cn=g6,o=ibm
objectclass: container
objectclass: ibm-dynamicGroup
memberURL: ldap:///o=ibm??sub?(cn=*3)

dn: cn=u1,o=ibm
objectclass: person
cn: u1
sn: user
userpassword: secret1
```

```
dn: cn=u2,o=ibm
objectclass: person
cn: u2
sn: user
userpassword: secret2

dn: cn=u3,o=ibm
objectclass: person
aclEntry: access-id:cn=u1,o=ibm:normal:rsc:system:rsc:at.cn:deny:s
aclEntry: access-id:cn=u2,o=ibm:normal:rsc:system:rsc:at.ibm-allGroups:deny:r
aclEntry: group:cn=ANYBODY:normal:rsc:system:rsc
cn: u3
sn: user
userpassword: secret3

dn: cn=u4,o=ibm
objectclass: person
aclentry: group:cn=ANYBODY:normal:rsc:system:rsc
aclentry: access-id:cn=u3,o=ibm:normal:rsc:system:rsc:at.ibm-allGroups:deny:r
cn: u4
sn: user
userpassword: secret4

dn: cn=u5,o=ibm
objectclass: person
cn: u5
sn: user
userpassword: secret5

dn: cn=u6,o=ibm
objectclass: person
cn: u6
sn: user
userpassword: secret6
```

Note: The **ibm-allMembers** and **ibm-allGroups** attributes are system class attributes. The **member** and **cn** attributes are normal class attributes.

ibm-allGroups and ibm-allMembers search and comparison examples:

Example 1: This example shows an **ibm-allMembers** attribute search on a static group entry.

```
ldapsearch –L –D "cn=u6,o=ibm" –w secret –b "cn=g4,o=ibm" "objectclass=*" ibm-allMembers

dn: cn=g4,o=ibm
ibm-allmembers: cn=u5,o=ibm
```

Access checking done for cn=u6,o=ibm:

1. Read access to the **ibm-allMembers** attribute in cn=g4,o=ibm.
2. Read access to the **member** attribute in cn=g4,o=ibm.

Example 2: This example shows an **ibm-allMembers** attribute search on a dynamic group entry.

```
ldapsearch –L –D "cn=u6,o=ibm" –w secret –b "cn=g5,o=ibm" "objectclass=*" ibm-allMembers

dn: cn=g5,o=ibm
ibm-allmembers: cn=u3,o=ibm
ibm-allmembers: cn=u4,o=ibm
```

Access checking done for u6,o=ibm:

1. Read access to the **ibm-allMembers** attribute in cn=g5,o=ibm.
2. Search access to the **cn** attribute in the returned entries, cn=u3,o=ibm and cn=u4,o=ibm, from the search filter specified in the **memberURL** attribute.

Note: **memberURL** attribute access rights do not matter.

Example 3: This example shows an **ibm-allMembers** attribute search on a nested group entry.

```
ldapsearch –L –D "cn=u6,o=ibm" –w secret –b "cn=g3,o=ibm" "objectclass=*" ibm-allMembers
dn: cn=g3,o=ibm
ibm-allmembers: cn=u3,o=ibm
ibm-allmembers: cn=u4,o=ibm
ibm-allmembers: cn=u5,o=ibm
```

Access checking done for cn=u6,o=ibm:

1. Read access to the **ibm-allMembers** attribute in cn=g3,o=ibm.
2. Read access to the **member** attribute in cn=g4,o=ibm.
3. Search access to the **cn** attribute in the returned entries, cn=u3,o=ibm and cn=u4,o=ibm, from the search filter specified in the **memberURL** attribute of cn=g5,o=ibm.
4. Search access to the **cn** attribute in the returned entry, cn=u3,o=ibm, from the search filter specified in the **memberURL** attribute of cn=g6,o=ibm.

   Note: Since cn=u3,o=ibm has already been added as an **ibm-allMembers** attribute value, a duplicate value will not be added.

Note: ibm-memberGroup access rights do not matter.

Example 4: This example shows an **ibm-allMembers** attribute search on a dynamic group entry when the bound user is not granted read access to the **ibm-allMembers** attribute.

```
ldapsearch -L -D "cn=u3,o=ibm"-w secret -b "cn=g5,o=ibm" "objectclass=*" ibm-allmembers
dn: cn=g5,o=ibm
```

Access checking done for cn=u3,o=ibm:

1. Read access to the **ibm-allMembers** attribute in cn=g5,o=ibm has been denied. Therefore, no **ibm-allMembers** attribute values will be added.

Example 5: This example shows an **ibm-allMembers** attribute search on a static group entry when the bound user does not have read authority on the **member** attribute.

```
ldapsearch -L -D "cn=u2,o=ibm" -w secret -b "cn=g2,o=ibm" "objectclass=*" ibm-allmembers
dn: cn=g2,o=ibm
```

Access checking done for cn=u2,o=ibm:

1. Read access to the **ibm-allMembers** attribute in cn=g2,o=ibm.
2. Read access to the **member** attribute in cn=g2,o=ibm has been denied. Therefore, the **member** attribute value will not be added as an **ibm-allMembers** attribute value.

Example 6: This example shows an **ibm-allMembers** attribute search on a dynamic group entry when the bound user does not have search authority in the entries that are to be returned for the attributes that are specified in the dynamic group filter.

```
ldapsearch -L -D "cn=u1,o=ibm" -w secret -b "cn=g5,o=ibm" "objectclass=*" ibm-allmembers
dn: cn=g5,o=ibm
ibm-allmembers: cn=u4,o=ibm
```

Access checking done for cn=u1,o=ibm:

1. Read access to the **ibm-allMembers** attribute in cn=g5,o=ibm.
2. Search access to the **cn** attribute in the returned entries, cn=u3,o=ibm and cn=u4,o=ibm, from the search filter specified in the **memberURL** attribute. However, search access has been denied on the **cn** attribute of cn=u3,o=ibm therefore it is not added as an **ibm-allMembers** attribute value.

Example 7: This example shows an **ibm-allMembers** comparison operation on a dynamic group entry.

```
ldapcompare -D "cn=u3,o=ibm" -w secret "cn=g5,o=ibm" "ibm-allmembers=cn=u3,o=ibm"
ldap_compare: Compare true
```

Note: Although an **ldapcompare** utility is not shipped with the z/OS LDAP client, some LDAP client providers on other platforms do ship an **ldapcompare** utility.

Access checking done for cn=u3,o=ibm:

1. Compare access to the **ibm-allMembers** attribute in cn=g5,o=ibm.
2. Search access to the **cn** attribute on the returned entries, cn=u3,o=ibm and cn=u4,o=ibm, from the search filter specified in the **memberURL** attribute.

Example 8: This example shows an **ibm-allGroups** attribute search where the user belongs to dynamic and nested group entries.

```
ldapsearch -L -D "cn=u6,o=ibm" -w secret -b "cn=u4,o=ibm" "objectclass=*" ibm-allGroups
dn: cn=u4,o=ibm
ibm-allgroups: cn=g5,o=ibm
ibm-allgroups: cn=g3,o=ibm
ibm-allgroups: cn=g1,o=ibm
```

Access checking done for cn=u6,o=ibm:

1. Read access to the **ibm-allGroups** attribute in cn=u4,o=ibm.
2. Search access on the **cn** attribute in cn=u4,o=ibm from the search filter specified in the **memberURL** attribute in cn=g5,o=ibm.

Since cn=g3,o=ibm has cn=g5,o=ibm as an **ibm-memberGroup** attribute value, cn=g3,o=ibm is added as an **ibm-allGroups** attribute as well. cn=g1,o=group has cn=g3,o=ibm as an **ibm-memberGroup** value, therefore cn=g1,o=group is also added as an **ibm-allGroups** attribute value.

Example 9: This example shows an **ibm-allGroups** attribute search where the user belongs to static and nested group entries.

```
ldapsearch -L -D "cn=u1,o=ibm" -w secret -b "cn=u2,o=ibm" "objectclass=*" ibm-allGroups
dn: cn=u2,o=ibm
ibm-allgroups: cn=g2,o=ibm
ibm-allgroups: cn=g1,o=ibm
```

Access checking done for cn=u1,o=ibm:

1. Read access to the **ibm-allGroups** attribute in cn=u2,o=ibm.
2. Read access to the **member** attribute in cn=g2,o=ibm.

Since cn=g1,o=ibm has an **ibm-memberGroup** attribute value of cn=g2,o=ibm, cn=g1,o=ibm is added as an **ibm-allGroups** attribute value.

Example 10: This example shows an **ibm-allGroups** attribute search where the user being searched belongs to static and nested group entries. The bound user has read authority to the **ibm-allGroups** attribute of the user being searched, but does not have read authority on the **member** attribute in the static group entry.

```
ldapsearch -L -D "cn=u2,o=ibm" -w secret -b "cn=u2,o=ibm" "objectclass=*" ibm-allGroups
dn: cn=u2,o=ibm
```

Access checking done for cn=u2,o=ibm:

1. Read access to the **ibm-allGroups** attribute in cn=u2,o=ibm.
2. Read access to the **member** attribute of cn=g2,o=ibm is denied. Therefore, cn=g2,o=ibm is not added as an **ibm-allGroups** attribute value.

Example 11: This example shows an **ibm-allGroups** search where the bound user does not have read authority on the **ibm-allGroups** attribute.

```
ldapsearch -L -D "cn=u3,o=ibm" -w secret -b "cn=u4,o=ibm" "objectclass=*" ibm-allGroups
dn: cn=u4,o=ibm
```

Access checking done for cn=u3,o=ibm:

1. Read access to the **ibm-allGroups** attribute in cn=u4,o=ibm is denied. Therefore, no **ibm-allGroups** attribute values are added.

Example 12: This example shows an **ibm-allGroups** comparison operation where the bound user is allowed to determine that a user belongs to a nested group entry.

```
ldapcompare -D "cn=u2,o=ibm" -w secret "cn=u3,o=ibm" "ibm-allGroups=cn=g1,o=ibm"
ldap_compare: Compare true
```

Note: Although an **ldapcompare** utility is not shipped with the z/OS LDAP client, some LDAP client providers on other platforms do ship an **ldapcompare** utility.

Access checking done for cn=u2,o=ibm:

1. Compare access to the **ibm-allGroups** attribute in cn=u3,o=ibm.
2. Search access to the **cn** attribute of cn=u3,o=ibm is granted from the search filter specified in the **memberURL** attribute in cn=g5,o=ibm.

Since cn=g3,o=ibm has cn=g5,o=ibm as an **ibm-memberGroup** attribute value, cn=g3,o=ibm is added as an **ibm-allGroups** attribute as well. cn=g1,o=group has cn=g3,o=ibm as an **ibm-memberGroup** value, therefore cn=g1,o=group is also added as an **ibm-allGroups** attribute value. Therefore, the compare operation will return an LDAP_COMPARE_TRUE to the client application.

Chapter 22. Using access control

Access control of information in the LDAP server is specified by setting up Access Control Lists (ACLs). TDBM and GDBM ACLs provide a means to protect information stored in an LDAP directory. Administrators use ACLs to restrict access to different portions of the directory, or specific directory entries. LDAP directory entries are related to each other by a hierarchical tree structure. Each directory entry (or object), contains the entry's distinguished name, a set of attributes and their corresponding values. When using the TDBM or GDBM backend, ACLs are created and managed using the **ldap_add** and **ldap_modify** APIs. ACLs can also be entered using the **ldif2tdbm** utility (TDBM only).

ACLs are represented by a set of attributes which appear to be a part of the entry. The attributes associated with access control, such as **entryOwner**, **ownerPropagate**, **aclEntry**, and **aclPropagate**, are unusual in that they are logically associated with each entry, but can have values which depend upon other entries higher in the directory hierarchy. Depending upon how they are established, these attribute values can be explicit to an entry, or inherited from an ancestor entry.

Use of LDAP's SDBM backend allows a user to be authenticated to the directory namespace using the RACF ID and password. The RACF identity becomes associated with the user's RACF-style distinguished name that was used on the LDAP bind operation. It is then possible to set up ACLs for entries managed by the TDBM or GDBM backend using RACF-style user and group DNs. This controls access to TDBM or GDBM database directory entries using the RACF user or group identities.

Access control attributes

Access to LDAP directory entries and attributes is defined by Access Control Lists (ACLs). Each entry in the directory contains a special set of attribute/value pairs which describe who is allowed to access information within that entry. Table 43 shows the set of attributes which are related to access control. More in-depth information about each attribute is given following the table.

Starting in z/OS V1R4, it is possible to specify access control settings for individual attribute types. This is called attribute-level access control. Also, added in z/OS V1R4 is the ability to explicitly deny access to information. In prior releases, all ACL information indicated what access was granted with the implication that access to all other information was denied.

Table 43. TDBM ACL and entry owner attributes

ACL attributes

aclEntry	This is a multi-valued attribute that contains the names and permissions associated with those names that have access to information in the directory entry (or the entry along with the subtree of information below the entry, depending on the setting of the **aclPropagate** attribute).
aclPropagate	This is a single-valued boolean attribute which indicates whether the **aclEntry** information applies only to the directory entry it is associated with or to the entire subtree of information including and below the directory entry it is associated with. Note that propagation does not apply to entries that have an explicit **aclEntry** defined for the entry and that propagation stops at the next propagating ACL (`aclPropagate=TRUE`) that is encountered in the directory subtree.
aclSource	This is a single-valued attribute that is not modifiable. This attribute is managed by the directory server and cannot be changed by the **ldapmodify** command. This attribute, accessible for any directory entry, indicates the distinguished name of the entry that holds the ACL that applies to the entry. This attribute is useful in determining which propagating ACL is used to control access to information in the directory entry.

Entry owner attributes

Table 43. TDBM ACL and entry owner attributes (continued)

entryOwner	This is a multi-valued attribute that contains the distinguished names of users or groups that are considered owners of the directory entry (or the entry along with the subtree of information below the entry, depending on the setting of the **ownerPropagate** attribute).
ownerPropagate	This is a single-valued boolean attribute which indicates whether the **entryOwner** information applies only to the directory entry it is associated with or to the entire subtree of information including and below the directory entry it is associated with. Note that propagation does not apply to entries that have an explicit **entryOwner** defined for the entry and that propagation stops at the next propagating **entryOwner** (`ownerPropagate=TRUE`) that is encountered in the directory subtree.
ownerSource	This is a single-valued attribute that is not modifiable. This attribute is managed by the directory server and cannot be changed by the **ldapmodify** command. This attribute indicates the distinguished name of the entry that holds the **entryOwner** that applies to the entry. This attribute is useful in determining which propagating **entryOwner** is used to control access to information in the directory entry.

aclEntry attribute

An **aclEntry** is a multi-valued attribute which contains information pertaining to the access allowed to the entry and each of its attributes. An **aclEntry** lists the following types of information:

- Who has rights to the entry (scope of the protection). Also called the subject.
- What attributes, and classes of attributes (attribute access classes) that the subject has access to.
- What rights the subject has (permissions to attribute and classes of attributes).

Scope of protection

The scope of the protection is based on the following three types of privilege attributes:

access-id
: The distinguished name of an entry being granted access.

group The distinguished name of the group entry being granted access.

role The distinguished name of the group entry being granted access. (Use this for the TDBM backend only.)

Access control groups can be either static, dynamic, or nested groups. The following object classes are evaluated as group entries when the TDBM DB_VERSION is at least 3.0: **ibm-staticGroup, groupOfNames, groupOfUniqueNames, accessRole, accessGroup, ibm-dynamicGroup, groupOfUrls,** and **ibm-nestedGroup**. See Chapter 21, “Static, dynamic, and nested groups,” on page 257 for additional information on static, dynamic, and nested groups.

Privilege attributes take the form of *type*:*name* where *type* refers to either **access-id**, **group**, or **role** and *name* is the distinguished name.

Note: The distinguished name that is used need not be the name of an entry in the directory. The distinguished name is the name that represents the user that has authenticated to the directory server.

The *type*: portion of this clause is optional.

Examples

In this example, the DN type is access-id and the DN itself is cn=personA, ou=deptXYZ, o=IBM, c=US.

```
access-id:cn=personA, ou=deptXYZ, o=IBM, c=US
```

In this example, the DN type is group and the DN itself is cn=deptXYZRegs, o=IBM, c=US.

```
group:cn=deptXYZRegs, o=IBM, c=US
```

This is an example of how to use the RACF identity established with SDBM in a TDBM ACL.

```
access-id:racfid=YourID,profileType=user,sysplex=YourSysplex
group:racfid=YourGroup,profileType=group,sysplex=YourSysplex
```

Attribute access classes

Attributes requiring similar permission for access are grouped together in classes. Attributes are assigned to an attribute access class within the schema definitions. The **ibmAttributeTypes** attribute of the TDBM and GDBM schema entry holds the attribute type's access class. The three attribute access classes are:

- **normal**
- **sensitive**
- **critical**

Each of these attribute access classes is discrete. If a user has write permission to **sensitive** attributes, then the user does not automatically have write permission to **normal** attributes. This permission must be explicitly defined.

The default attribute access class for an attribute is **normal**. By default, all users have read access to **normal** attributes. There are two additional attribute access classes used internally by LDAP for system attributes. These attribute access classes are **restricted** and **system**. You can specify these access classes when granting permissions in ACLs.

For example, a person's name would typically be defined in the **normal** class. Perhaps a social security number would be considered **sensitive**, and any password information for the user would be considered **critical**. Following are some example definitions excerpted from **schema.user.ldif**. Note that the attribute **userPassword** is defined with access class **critical**.

```
attributetypes: (
  2.5.4.49
  NAME ( 'dn'  'distinguishedName'  )
  DESC 'This attribute type is not used as the name of the object itself,
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  USAGE userApplications
  )
ibmattributetypes: (
  2.5.4.49
  ACCESS-CLASS normal
  )

attributetypes: (
  2.5.4.35
  NAME 'userPassword'
  DESC 'Holds a password value for a distinguished name.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
  USAGE userApplications
  )
ibmattributetypes: (
  2.5.4.35
  ACCESS-CLASS critical
  )
```

After z/OS V1R4, it is possible to specify access controls on individual attributes. However, when defining schema an access class is always defined for the attribute type. If not specified, that attribute type is defined to belong to the **normal** class.

Access permissions

Following is the set of access permissions.

Table 44. Permissions which apply to an entire entry

Add	Add an entry below this entry

Table 44. Permissions which apply to an entire entry (continued)

Delete	Delete this entry

Table 45. Permissions which apply to attribute access classes

Read	Read attribute values
Write	Write attribute values
Search	Search filter can contain attribute type
Compare	Compare attribute values

Syntax

Following is the **aclEntry** attribute value syntax:

[**access-id:**|**group:**|**role:**]*subject_DN*[*granted_rights*]

The *subject_DN* is any valid DN which represents the object (entry) to which privileges are being granted. The DN ends when the first granted rights keyword is detected.

The *granted_rights* is specified as follows where *object_rights_list* is one or more elements of the set [**ald**], and *attr_rights_list* is one or more elements of the set [**r|w|s|c**].

[**:object:**[**grant:**|**deny:**]*object_rights_list*] [**:normal:**[**grant:**|**deny:**]*attr_rights_list*]
[**:sensitive:**[**grant:**|**deny:**]*attr_rights_list*] [**:critical:**[**grant:**|**deny:**]*attr_rights_list*]
[**:restricted:**[**grant:**|**deny:**]*attr_rights_list*] [**:system:**[**grant:**|**deny:**]*attr_rights_list*]
[**:at.***attr_name***:[grant:**|**deny:]***attr_rights_list*]

Note: The **restricted** attributes are: **aclEntry**, **aclPropagate**, **entryOwner**, and **ownerPropagate**. In order to update access control information, you must have permissions to read and write these attributes. The **system** attributes include **aclSource** and **ownerSource** and other attributes for which the server controls the values. In order to update access control information, you must have permission to read and write these attributes.

Following are some examples of valid **aclEntry** values:

```
access-id:cn=Tim, o=Your Company:normal:rwsc:sensitive:rsc:object:ad

role:cn=roleGroup, o=Your Company:object:ad:normal:rsc:sensitive:rsc

group:cn=group1, o=Your Company:system:csr:normal:sw

cn=Lisa, o=Your Company:normal:rwsc:sensitive:rwsc:critical:rwsc:restricted:rwsc:system:rwsc

cn=Ken, o=Your Company:normal:rsc

group:cn=group2,dc=yourcompany,dc=com:normal:rwsc:at.cn:deny:w:sensitive:grant:rsc

cn=Karen,dc=yourcompany,dc=com:at.cn:grant:rwsc:normal:deny:rwsc

cn=Mary,dc=yourcompany,dc=com:normal:rwsc:sensitive:rwsc:critical:deny:rwsc:at.userpassword:w
```

The access control implementation supports several “pseudo-DNs”. These are used to refer to large numbers of subject DNs which, at bind time, share a common characteristic in relation to either the operation being performed or the target object on which the operation is being performed. Currently, three pseudo DNs are defined:

```
group:cn=anybody
group:cn=authenticated
access-id:cn=this
```

The `group:cn=anybody` refers to all subjects, including those that are unauthenticated (considered anonymous users). All users belong to this group automatically. The `group:cn=authenticated` refers to any

DN which has been authenticated to the directory. The method of authentication is not considered. The `access-id:cn=this` refers to the bind DN which matches the target object's DN on which the operation is performed.

See "Access determination" on page 275 for information on how the **aclEntry** values are used to determine access.

aclPropagate attribute

Each entry with an explicit ACL has associated with it an **aclPropagate** attribute. By default, the entry's explicit ACL is inherited down the hierarchy tree, and its **aclPropagate** attribute is set to **TRUE**. If set to **FALSE**, the explicit ACL for the entry becomes an override, pertaining only to the particular entry. The **aclPropagate** syntax is Boolean. See "Propagating ACLs" on page 277 for more information.

aclSource attribute

Each entry has an associated **aclSource**. This reflects the DN with which the ACL is associated. This attribute is kept and managed by the server, but may be retrieved for administrative purposes. This attribute cannot be set, only retrieved.

In order to aid in migration from RDBM databases to TDBM databases, LDIF files can contain **aclSource** values for certain entries. The rule is as follows:

> If **aclSource** is specified, the value must be the same as the distinguished name of the entry in which it is specified. If **aclSource** is specified and has a different value than the distinguished name of the entry within which it appears, the add and load of that entry will fail.

The derivation of **aclSource** is further explained in "Propagating ACLs" on page 277.

entryOwner attribute

Each entry has an associated **entryOwner**. The **entryOwner** might be a user or a group, similar to what is allowed within the **aclEntry**. However, the **entryOwner** subject has certain privileges over the entry.

Entry owners are, in essence, the administrators for a particular entry. They have full access on that particular entry, similar to the administrator DN. Note that the administrator DN has full permission on every entry in the database. Each **entryOwner** attribute value is a distinguished name. However, for compatibility with previous releases, the distinguished name can be preceded with **access-id:**, **group:**, or **role:**.

Note: The distinguished name that is used need not be the name of an entry in the directory. The distinguished name is the name that represents the user that has authenticated to the directory server.

Entry owners are not constrained by permissions given in the **aclEntry**. They have complete access to any entry attribute, and can add and delete entries as desired.

Entry owners, the administrator DN, and users who have write permission for restricted attributes are the only people who are allowed to change the attributes related to access control.

ownerPropagate attribute

Owner propagation works exactly the same as ACL propagation. By default, owners are inherited down the hierarchy tree, and their owner propagate attribute is set to **TRUE**. If set to **FALSE**, the owner becomes an override, pertaining only to the particular entry. The **ownerPropagate** syntax is boolean.

ownerSource attribute

Each entry also has an associated **ownerSource**. This reflects the DN with which the owner values are associated. This attribute is kept and managed by the server, but can be retrieved for administrative purposes. This attribute cannot be set, only retrieved.

In order to support existing ACL support from the RDBM backend and aid in migration from RDBM databases to TDBM databases, LDIF files can contain **ownerSource** values for certain entries. The rule is as follows:

> If **ownerSource** is specified, the value must be the same as the distinguished name of the entry in which it is specified. If **ownerSource** is specified and has a different value than the distinguished name of the entry within which it appears, the add and load of that entry will fail.

Initializing ACLs with TDBM

The TDBM backend adds an ACL to the ″suffix entry″ if no **aclEntry** value is specified during the add of this entry (whether the add was done using **ldapadd** or **ldif2tdbm**). This improves performance of future ACL modifications made to an ACL placed on the suffix entry. The ACL that is used is:

```
aclEntry: cn=anybody:normal:rsc:system:rsc
aclPropagate: TRUE
```

Similarly, if no entry owner is specified when the suffix entry is created, **entryOwner** is added to the entry with a value set to the administrator DN, along with **ownerPropagate TRUE**.

Default ACLs with TDBM

Every entry must have an ACL. If there is no ACL explicitly specified in the entry and no parent entry is propagating its ACL, then a default ACL is assigned to the entry. The default ACL is treated differently than a normal **aclEntry** value. The default value cannot be deleted. If an **aclEntry** value is later added to the entry, explicitly or by inheritance, the entire default **aclEntry** value is replaced. The LDAP server sets the value of the **aclSource** attribute to 'default' when the entry is using the default ACL. The default ACL is:

```
aclentry: access-id:CN=ADMIN:normal:rwsc:sensitive:rwsc:critical:rwsc:restricted:rwsc:system:rwsc
aclentry: group:CN=ANYBODY:normal:rsc:system:rsc
aclentry: group:CN=AUTHENTICATED:normal:rsc:system:rsc
```

Similarly, every entry must have an entry owner. If none is specified or inherited, a default **entryOwner** value set to the administrator DN is assigned to the entry. The default value cannot be deleted. If an **entryOwner** value is later added to the entry, explicitly or by inheritance, the entire default **entryOwner** value is replaced. The LDAP server sets the value of the **ownerSource** attribute to 'default' when the entry is using the default owner.

Initializing ACLs with GDBM

When the LDAP sever is started with GDBM configured for the first time, the LDAP server creates the change log suffix entry, cn=changelog. The suffix entry is created with an **aclEntry** and **entryOwner** value that allows access only to the LDAP administrator and propagates the **aclEntry** and **entryOwner** values. The **aclEntry** and **entryOwner** values in the suffix can be modified, but these attributes cannot be entirely removed from the suffix entry and they cannot be changed to be non-propagating. In other words, the change log suffix entry always contains propagating **aclEntry** and **entryOwner** values. If desired, different ACL values can be placed on specific change log entries to override the inherited values from the change log suffix entry.

Access determination

The same distinguished name (DN) may be granted different access permissions to an entry, from specific access permissions to the DN and from group memberships (including the authenticated and anybody groups). The LDAP server uses the following algorithm to determine which permissions to grant a DN based on the values in the **aclEntry** attribute:

- if there is a specific value for the DN, the DN gets those permissions only
- else if there is a cn=this value and the DN is the distinguished name of the entry, the DN gets those permissions only
- else if there are one or more group values that the DN is a member of, the DN gets the union of the permissions for those groups
- else if there is a cn=authenticated value and the DN is authenticated to the directory with an LDAP bind operation, the DN gets those permissions only
- else if there is a cn=anybody value, the DN gets those permissions only
- otherwise the DN gets no permissions

Each of the LDAP access permissions is discrete. One permission does not imply another.

With the support added in z/OS V1R4 for attribute-level permissions as well as grant/deny support, the order of evaluation of the separate permissions clauses is important. The access control permissions clauses are evaluated in a precedence order, not in the order in which they are found in the ACL entry value. With the new support, there are four types of permissions settings: access-class grant permissions, access-class deny permissions, attribute-level grant permissions, and attribute-level deny permissions. The precedence for these types of permissions is as follows (from highest precedence to lowest):

- attribute-level deny permissions
- attribute-level grant permissions
- access-class deny permissions
- access-class grant permissions

Using this precedence, a deny permission takes precedence over a grant permission (for the same item specified) while attribute-level permissions take precedence over access-class permissions.

Following are examples for permissions:

Example 1

```
aclentry: group:cn=Anybody:normal:rsc
```

In this example, unauthenticated (anonymous) users have permission to read, search and compare all attributes within the **normal** attribute access class. ACL entry values for unauthenticated users use pseudoDN `cn=Anybody`.

Example 2

```
aclentry: access-id:cn=personA,ou=deptXYZ,o=IBM,c=US:object:ad:normal:rwsc:sensitive:rwsc:critical:rsc
```

In this example, the user corresponding to `cn=personA, ou=deptXYZ, o=IBM, c=US` has permission to add entries below the entry, to delete the entry, to read, write, search and compare both **normal** and **sensitive** attributes, and to read, search and compare **critical** attributes.

Example 3

```
aclentry: group:cn=Authenticated:normal:rwsc:sensitive:rwsc
```

In this example, users who have authenticated to the directory where a specific **aclEntry** value does not apply, will be allowed to read, write, search, and compare, **normal** and **sensitive** attributes in the directory entry.

Example 4

```
aclEntry: cn=Tim,dc=yourcompany,dc=com:at.cn:deny:w:normal:rwsc
```

In this example, `cn=Tim,dc=yourcompany,dc=com` is granted read, write, search, and compare to **normal** attributes except for the **cn** attribute in which write access is denied. Note that the following ACL entry results in the same access:

```
aclEntry: cn=Tim,dc=yourcompany,dc=com:normal:rwsc:at.cn:deny:w
```

The evaluation of the permissions clauses is based on precedence, not order in the ACL entry value(s).

Example 5

```
aclEntry: cn=Karen,dc=yourcompany,dc=com:normal:rwsc:sensitive:rsc:at.userpassword:w:
 critical:deny:rwsc
```

In this example, `cn=Karen,dc=yourcompany,dc=com` is granted read, search, and compare to **normal** and **sensitive** attributes, and write to **normal** attributes and the **userpassword** attribute. All access to **critical** attributes (except for write in **userpassword**) is turned off.

Example 6

```
aclEntry: group:cn=group1,dc=yourcompany,dc=com:normal:rwsc
aclEntry: group:cn=group2,dc=yourcompany,dc=com:sensitive:rwsc:at.cn:deny:w
```

In this example, a member of group1 only would be granted read, write, search, and compare to **normal** attributes. A member of both group1 and group2 would be granted read, write, search, and compare to **normal** and **sensitive** attributes, excluding write access to the **cn** attribute. This is an example where a member of both groups is granted access to less information than what is granted to each of the two groups individually.

Example 7

```
aclEntry: access-id:cn=Tim,dc=yourcompany,dc=com:normal:rwsc:at.cn:rsc
```

In this example, `cn=Tim,dc=yourcompany,dc=com` is granted read, write, search, and compare on **normal** attributes and read, search, and compare on the **cn** attribute. Note that `cn=Tim,dc=yourcompany,dc=com` will also have write access to the **cn** attribute by virtue of **cn** having an access class of **normal**.

Example 8

```
aclEntry: access-id:cn=Tim,dc=yourcompany,dc=com:normal:rwsc:at.cn:deny:rsc
```

In this example, `cn=Tim,dc=yourcompany,dc=com` is granted read, write, search, and compare on **normal** attributes and denied read, search, and compare on the **cn** attribute. Note that `cn=Tim,dc=yourcompany,dc=com` will still have write access to the **cn** attribute by virtue of **cn** having an access class of **normal**.

Attribute classes and searching

In order to read information from the directory, the user must have read permission for the attribute.

Filter

In order to use an attribute in a search filter supplied on a search operation, the user must have search permission for the attribute.

Compare

In order to perform a compare operation on an attribute/value combination, the user must have compare permission on the attribute.

Requested attributes

If the user has the search permission on all attributes contained in the filter, the server returns as much information as possible. All requested attributes that the user has read permission for are returned.

For example, let the **aclEntry** be

```
group:cn=Anybody:normal:rsc:sensitive:c:critical:c
```

and let a client perform an anonymous search

```
ldapsearch -b "c=US" "cn=LastName" title userpassword telephoneNumber
```

where `title` is a **normal** attribute, `telephoneNumber` is a **sensitive** attribute, and `userpassword` is a **critical** attribute. See the *z/OS Integrated Security Services LDAP Client Programming* for more information about the **ldapsearch** utility.

Users performing anonymous searches are given the permission granted to the **cn=Anybody** group. In this example, permission exists to the filter since **cn** is in the **normal** attribute access class, and **cn=Anybody** has **s** (search) permission to the **normal** attribute access class. What is returned however, is only the **title** attribute for any matching entry. The **telephoneNumber** and **userPassword** attributes are not returned since **cn=Anybody** does not have read permissions on the **sensitive** and **critical** attribute access classes.

Propagating ACLs

ACLs can be set on any entry in the hierarchy. LDAP ACLs can propagate down through the directory hierarchy. These ACLs, called propagating ACLs, have the **aclPropagate** attribute set to **TRUE**. All descendents of this entry will inherit the ACL set at that point, unless overridden. In order to specify an ACL different from that of its parent, a new ACL must be explicitly set.

When setting the new ACL, there is again a choice of whether to propagate the ACL. If set to **TRUE**, the ACL will propagate down to all descendants. If set to **FALSE**, the ACL is not propagated; it instead becomes an override ACL. The ACL is not propagated down through the hierarchy, but instead applies only to the one particular entry that it is associated with within the hierarchy. If unspecified, **aclPropagate** is set to **TRUE**.

An entry without an explicit ACL receives its ACL from the nearest propagating ancestor ACL. Propagated ACLs do not accumulate as the depth in the tree increases. The scope of a propagated ACL is from the explicitly-set propagating ACL down through the tree until another explicitly-set propagating ACL is found.

Example of propagation

Following is the explicit ACL for entry `ou=deptXYZ, o=IBM, c=US` :

```
aclPropagate:  TRUE
aclEntry: group:cn=deptXYZRegs, o=IBM, c=US:normal:rcs:sensitive:rsc
aclEntry: access-id:cn=personA, ou=deptXYZ, o=IBM, c=US:object:ad:normal:rwsc:sensitive:rwsc:critical:rsc
aclEntry: group:cn=Anybody:normal:rsc
aclSource: ou=deptXYZ, o=IBM, c=US
```

In the absence of an explicit ACL for entry `cn=personA, ou=deptXYZ, o=IBM, c=US`, the following is the implicit, propagated ACL for the entry:

```
aclPropagate: TRUE
aclEntry: group:cn=deptXYZRegs, o=IBM, c=US:normal:rcs:sensitive:rsc
aclEntry: access-id:cn=personA, ou=deptXYZ, o=IBM, c=US:object:ad:normal:rwsc:sensitive:rwsc:critical:rsc
aclEntry: group:cn=Anybody:normal:rsc
aclSource:ou=deptXYZ, o=IBM, c=US
```

In this example, a propagating ACL has been set on `ou=deptXYZ, o=IBM, c=US`. No ACL has been set on the descendant `cn=personA, ou=deptXYZ, o=IBM, c=US`. Therefore, the descendant inherits its ACL value from the nearest ancestor with a propagating ACL. This happens to be `ou=deptXYZ, o=IBM, c=US`, which is reflected in the **aclSource** attribute value. The **aclEntry** and **aclPropagate** values are identical to those values in the explicit propagating ACL set at `ou=deptXYZ, o=IBM, c=US`.

Examples of overrides

Following is an explicit ACL for entry `o=IBM, c=US`:

```
aclPropagate:  TRUE
aclEntry:   group:cn=IBMRegs, o=IBM, c=US:normal:rcs:sensitive:rsc
aclEntry:   group:cn=Anybody:normal:rsc
aclSource:  o=IBM, c=US
```

Following is an explicit ACL for entry `ou=deptXYZ, o=IBM, c=US`:

```
aclPropagate: FALSE
aclEntry: group:cn=deptXYZRegs, o=IBM, c=US:normal:rcs:sensitive:rsc
aclEntry: access-id:cn=personA, ou=deptXYZ, o=IBM, c=US:object:ad:normal:rwsc:sensitive:rwsc:critical:rsc
aclEntry: group:cn=Anybody:normal:rsc
aclSource: ou=deptXYZ, o=IBM, c=US
```

Note that in the explicit ACLs above, **aclSource** is the same as the entry DN. This attribute is generated and managed by the directory server; it cannot be set when modifying ACLs.

Following is an implicit ACL for entry `cn=personA, ou=deptXYZ, o=IBM, c=US`:

```
aclPropagate:  TRUE
aclEntry:  group:cn=IBMRegs, o=IBM, c=US:normal:rcs:sensitive:rsc
aclEntry:  group:cn=Anybody:normal:rsc
aclSource:  o=IBM, c=US
```

In this example, a propagating ACL has been set on `o=IBM, c=US`. An override ACL has been set (**aclPropagate** is **FALSE**) on the descendant `ou=deptXYZ, o=IBM, c=US`. Therefore, the ACL set at `ou=deptXYZ, o=IBM, c=US` pertains only to that particular entry.

The descendant `cn=personA, ou=deptXYZ, o=IBM, c=US` inherits its ACL value from the nearest ancestor with a propagating ACL (which is `o=IBM, c=US` as reflected in the **aclSource**). The ACL on `ou=deptXYZ, o=IBM, c=US` is not used because **aclPropagate** is **FALSE**.

Other examples

In these examples, the administrator DN will be `cn=admin, c=US`.

The following example shows the default ACL:

```
aclPropagate:  TRUE
aclEntry: group:cn=Anybody:normal:rsc:system:rsc
aclSource: default
ownerPropagate:  TRUE
entryOwner:  access-id:cn=admin,c=US
ownerSource:  default
```

The following example shows a typical ACL for entry `cn=personA, ou=deptXYZ, o=IBM, c=US`:

```
aclPropagate:  TRUE
aclEntry:  group:cn=deptXYZRegs, o=IBM, c=US:normal:rcs:sensitive:rsc
aclEntry: access-id:cn=personA, ou=deptXYZ, o=IBM, c=US:object:ad:normal:rwsc:sensitive:rwsc:critical:rsc
```

```
aclEntry:  group:cn=Anybody:normal:rsc:system:rsc
aclSource:  ou=deptXYZ, o=IBM, c=US
ownerPropagate:  TRUE
entryOwner:  access-id:cn=deptXYZMgr, ou=deptXYZ, o=IBM, c=US
ownerSource:  ou=deptXYZ, o=IBM, c=US
```

This is an inherited ACL and an inherited owner. Both owner properties and ACL properties are inherited from entry `ou=deptXYZ, o=IBM, c=US`. In this example, members of group `cn=deptXYZRegs, o=IBM, c=US` have permission to read, search and compare entries in both the **normal** and **sensitive** attribute access classes. They do not have permission to add or delete entries under this entry. Nor do they have permission to access any information or change any information on attributes in the **critical** attribute access class. Unauthenticated, as well as all other bound users, have permission to read, search, and compare attributes in the **normal** and **system** attribute access classes only. The `personA` has add and delete permission on the entry; read, write, search, and compare permissions on **normal** and **sensitive** attributes; and read, search, and compare permission on **critical** attributes. The `deptXYZMgr` had full access to the entry since it is the owner of the entry. As always, the administrator also has unrestricted access to the entry.

Access control groups

Access control groups provide a mechanism for applying the same **aclEntry** attribute values to an entry for multiple users without having to create an explicit **aclEntry** for each user.

If the DB_VERSION of the TDBM backend is at least 3.0, the following object classes are evaluated as access control group entries: **accessGroup, accessRole, groupOfNames, groupOfUniqueNames, ibm-staticGroup, groupOfUrls, ibm-dynamicGroup,** and **ibm-nestedGroup**. Otherwise the only object class that is evaluated as an access control group entry is **accessGroup**. See Chapter 21, “Static, dynamic, and nested groups,” on page 257 for more information on static, dynamic, and nested groups.

When a user authenticates to the LDAP server, the groups which a user may belong to are first determined within the TDBM or SDBM backend to which the user successfully authenticates. If the user belongs to other groups in other TDBM backends, it is necessary to set the **extendedGroupSearching** configuration option for those backends so that the user’s membership in other groups is resolved. If a SASL EXTERNAL SSL certificate bind is done with a certificate DN that does not exist as an actual entry in any of the backends configured on the LDAP server but does exist as a member of a group, it is necessary to set the **extendedGroupSearching** configuration option for each backend where the certificate DN exists as a member of a group. This is necessary so that the group memberships of the certificate DN are resolved correctly. For more information, refer to Chapter 8, “Customizing the LDAP server configuration,” on page 53, specifically, the **extendedGroupSearching** configuration option, on page 62.

Retrieving ACL information from the LDAP server

In order to retrieve all of the ACL information in a namespace, use the **ldapsearch** command, as shown in the following example:

```
ldapsearch -h 127.0.0.1 -D "cn=admin, dc=Your Company,dc=com" -w xxxxxx -b "dc=Your Company,dc=com" \
"(objectclass=*)" aclEntry aclPropagate aclSource entryOwner ownerPropagate ownerSource
```

```
dn: dc=Your Company, dc=com
aclPropagate: TRUE
aclEntry: CN=ADMIN:normal:rwsc:sensitive:rwsc:critical:rwsc:object:ad
aclEntry: CN=ANYBODY:normal:rsc:system:rsc
aclSource: dc=Your Company, dc=com
ownerPropagate: TRUE
entryOwner: CN=ADMIN
ownerSource: default
```

This command performs a subtree search starting at the root of the tree (assuming the root of the tree is `"dc=Your Company,c=com"`) and returns the six ACL attributes for each entry in the tree. It is necessary to

specifically request the six ACL attributes because they are considered as “operational” and, therefore, can only be returned on a search if requested. (See IETF RFC 2251, *The Lightweight Directory Access Protocol (V3)*.)

ACL information (**aclEntry**, **aclPropagate**, **aclSource**, **entryOwner**, **ownerPropagate**, and **ownerSource**) is returned for all entries. For those entries that contain ACLs, the **aclSource** and **ownerSource** attributes contain the same DN as the entry DN. For those entries that do not contain ACLs, the **aclSource** and **ownerSource** attributes contain distinguished names of the entries that contain the ACL information (**aclEntry** and **entryOwner**) that are used for access control checking of information in that entry.

Notes:

1. It is possible for the **aclSource** and **ownerSource** attributes to contain the value **default**. This is not a distinguished name but rather represents that the ACL that applies to the entry is the default ACL (that is, no ACL information has been specified to apply to the entry).
2. If the tree is larger than the **sizeLimit** setting in the LDAP server configuration file or on the search command, then not all entries are returned. See the **sizeLimit** configuration option in Chapter 8, “Customizing the LDAP server configuration,” on page 53 for more information.

You can also use the same method to get the ACL information for a portion of the namespace by specifying the **-b** *searchbase* parameter on the **ldapsearch** command, where *searchbase* is the starting point for the search.

Creating and managing access controls

To create and update ACLs in the TDBM backend, use a tool implementing **ldap_modify** APIs, such as **ldapmodify**. The **ldapmodify** program is a general directory tool that allows creation, modification, and deletion of any set of attributes that are associated with an entry in the directory. Since access control information is maintained as a set of additional attributes within an entry, **ldapmodify** is a natural choice for administering access control information in the TDBM database backend.

See *z/OS Integrated Security Services LDAP Client Programming* for details on using the LDAP utilities, such as **ldapmodify**.

Creating an ACL

In order to create an ACL, the **aclEntry** and **aclPropagate** attributes must be added to the information stored for an entry. The **aclEntry** and **aclPropagate** attributes are added to an entry by either specifying them as part of the entry information when the entry is added to the directory or by modifying the entry after it exists to contain the **aclEntry** and **aclPropagate** information.

It is possible to create an ACL without specifying the **aclPropagate** attribute. In this case, the **aclPropagate** attribute is assumed to have a value of **TRUE** and is added into the directory entry automatically, along with the **aclEntry** information.

Since the **ldapmodify** command is very powerful, all the possible ways of adding the **aclEntry** and **aclPropagate** information cannot be shown here. The examples shown here describe the more common uses of the **ldapmodify** command to add ACL information.

Figure 39 on page 281 shows how to add a propagating ACL with three **aclEntry** values to an existing entry replacing any current **aclEntry** value.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f newAcl.ldif
```

Where `newAcl.ldif` contains:

```
dn: cn=tim, o=Your Company
changetype: modify
replace: aclentry
aclEntry: cn=jeanne, o=Your Company:
 normal:rsc:sensitive:rsc:critical:rsc
aclEntry: cn=jeff, cn=tim, o=Your Company:normal:rsc
aclEntry: cn=tim, o=Your Company:
 normal:rwsc:sensitive:rwsc:critical:rwsc
-
```

Figure 39. Example of adding propagating ACL to existing entry in directory

The ACL added in Figure 39 is created as a propagating ACL since the **aclPropagate** attribute is not specified and so assumed to be **TRUE**. This means that the ACL will apply to all entries below `cn=tim, o=Your Company` that do not already have an ACL associated with them. Note that the first and last **aclEntry** values span two lines in the `newAcl.ldif` file. In order to do this, the first character on the continued line must be a space character, as shown in the example.

While it is not required that the LDAP administrator update all ACL information, the examples in this section all use the administrator when updating ACLs. Further, the use of `-h 127.0.0.1` implies that the **ldapmodify** commands are performed from the same system on which the LDAP server is running and that the LDAP server is listening on TCP/IP port 389. Refer to the **ldapmodify** command description in *z/OS Integrated Security Services LDAP Client Programming* for more details on the **-h**, **-p**, **-D**, and **-w** command-line options. The ACL attributes can be updated from any LDAP client as long as the user performing the updates has the proper authorization to update the ACL information.

The ACL attributes are defined to be in a special access class called **restricted**. Thus, in order to allow someone other than the LDAP administrator to update the ACL attributes, they must either be the entry owner or have the proper authorization to **restricted** attributes. Figure 40 shows an example of adding an ACL so that `cn=jeanne, o=Your Company` can update the ACL information:

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f newAcl.ldif
```

Where `newAcl.ldif` contains:

```
dn: cn=jeanne, o=Your Company
changetype: modify
replace: aclentry
aclEntry: cn=jeanne, o=Your Company:
 normal:rsc:sensitive:rsc:critical:rsc:restricted:rwsc
aclEntry: cn=jeff, cn=tim, o=Your Company:normal:rsc
aclEntry: cn=tim, o=Your Company:
 normal:rsc
-
```

Figure 40. Example of adding propagating ACL to existing entry in the directory.

The ACL added in Figure 40 allows `cn=jeanne, o=Your Company` to update the ACL information for this entry. In addition, since the ACL is a propagating ACL, this allows `cn=jeanne, o=Your Company` to create new ACL information against any entry that is controlled by this ACL. Care must be taken here, however, since it is possible for `cn=jeanne, o=Your Company` to set up an ACL which then does not allow `cn=jeanne, o=Your Company` update capability on the ACL information. If this occurs, a user with sufficient authority (the administrator, for example) must be used in order to reset/change the ACL information.

Figure 41 shows an example of adding a non-propagating ACL. A non-propagating ACL applies only to the entry to which it is attached and not to the subtree of information that might be stored below the entry in the directory.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f newAcl.ldif
```

Where `newAcl.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
replace: aclentry
aclEntry: cn=tim, o=Your Company:normal:rwsc:sensitive:rwsc:
 critical:rwsc:restricted:rwsc
aclEntry: cn=jeff, cn=tim, o=Your Company:normal:rwsc:
 sensitive:rwsc:critical:rwsc
aclEntry: cn=jeanne, o=Your Company:normal:rsc
-
replace: aclpropagate
aclPropagate:FALSE
-
```

Figure 41. Example of setting up a non-propagating ACL

Setting up a non-propagating ACL is similar to setting up a propagating ACL. The difference is that the **aclPropagate** attribute value is set to **FALSE**.

Modifying an ACL

Once an ACL exists for an entry in the directory, it may have to be updated. To do this, the **ldapmodify** command is used. As described earlier in this chapter, while the **ldapmodify** command is used in these examples, what is really being used is an LDAP client application, issuing LDAP modify operations to the LDAP server. Thus, modifications to ACL information need not be performed from the same system on which the LDAP server is running.

Modifications to ACLs can be of a number of different types. The most common modifications are to:

- Add an additional **aclEntry** value to the ACL to allow another person or group access to the entry
- Change an ACL from propagating to non-propagating (not permitted for the GDBM change log suffix, cn=changelog)
- Remove an **aclEntry** value which exists in the ACL to disallow another person or group access to the entry that they had before.

Figure 42, Figure 43 on page 283, and Figure 44 on page 283 show examples of these modifications, respectively.

Figure 42 shows how an additional **aclEntry** value is added to existing ACL information.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modAcl.ldif
```

Where `modAcl.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
add: aclentry
aclEntry: cn=dylan, cn=tim, o=Your Company:
 normal:rwsc:sensitive:rwsc:critical:rwsc:restricted:rwsc
-
```

Figure 42. Example of adding an aclEntry attribute value

In Figure 42 on page 282, cn=dylan, cn=tim, o=Your Company is granted permissions against the cn=jeff, cn=tim, o=Your Company entry in the directory. The existing ACL information remains in the entry; the **aclEntry** attribute value for cn=dylan, cn=tim, o=Your Company is added to this information.

Figure 43 shows how to modify an existing ACL to be non-propagating instead of propagating.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modAcl.ldif
```

Where modAcl.ldif contains:

```
dn: cn=tim, o=Your Company
changetype: modify
replace: aclpropagate
aclPropagate: FALSE
-
```

Figure 43. Example of modifying aclPropagate attribute

In Figure 43, the existing ACL against cn=tim, o=Your Company is modified to be a non-propagating ACL instead of a propagating ACL. This means that the ACL will no longer apply to entries below cn=tim, o=Your Company in the directory tree. Instead, the first propagating ACL that is found in an entry above cn=tim, o=Your Company will be applied to the entries below cn=tim, o=Your Company. If no propagating ACL is found in the entries above cn=tim, o=Your Company, then the default ACL is used.

Figure 44 shows how to remove an **aclEntry** value from existing ACL information:

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modAcl.ldif
```

Where modAcl.ldif contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
delete: aclentry
aclEntry: cn=dylan, cn=tim, o=Your Company
-
```

Figure 44. Example of removing a single aclEntry attribute value

In Figure 44, the **aclEntry** attribute value for cn=dylan, cn=tim, o=Your Company is removed from the ACL information for entry cn=jeff, cn=tim, o=Your Company.

Deleting an ACL

In order to delete an ACL that is attached to an entry in the directory, the **aclEntry** and **aclPropagate** attributes must be deleted from the entry. To do this, use the **ldapmodify** command to delete the entire attribute (all values) from the entry.

Note: This is not allowed for the GDBM change log suffix entry, cn=changelog.

Figure 45 on page 284 shows an example of deleting an ACL from an entry.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f delAcl.ldif
```

Where `delAcl.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
delete: aclentry
-
delete: aclPropagate
-
```

Figure 45. Example of deleting an ACL from an entry

In Figure 45, the existing ACL against `cn=jeff, cn=tim, o=Your Company` is removed. This means that the ACL will no longer apply to the entry. Instead, the first propagating ACL that is found in an entry above `cn=jeff, cn=tim, o=Your Company` will be applied to `cn=jeff, cn=tim, o=Your Company`. If no propagating ACL is found in the entries above `cn=jeff, cn=tim, o=Your Company`, then the default ACL is used.

Note: Both the **aclEntry** and **aclPropagate** attributes must be deleted from the entry even though the **aclPropagate** attribute might not have been specified during the creation of the ACL information.

Creating an owner for an entry

In addition to the access control list control of directory entries, each entry can have assigned to it an entry owner or set of entry owners. As an entry owner, full access is allowed to the entry. Entry owners are granted add and delete permission, as well as read, write, search, and compare for all attribute classes. Entry owners can add and modify ACL information on the entries for which they are specified as the owner.

Entry owners are listed in the **entryOwner** attribute. Just like **aclEntry** information, **entryOwner** information can be propagating or non-propagating based on the setting of the **ownerPropagate** attribute. Like the **aclSource** attribute for **aclEntry** information, the **ownerSource** attribute lists the distinguished name of the entry that contains the **entryOwner** attribute which applies to the entry. The **ownerSource** attribute is set by the server and cannot be directly set when modifying the ACLs.

In order to create an entry owner, the **entryOwner** and **ownerPropagate** attributes must be added to the information stored for an entry. The **entryOwner** and **ownerPropagate** attributes are added to an entry by either specifying them as part of the entry information when the entry is added to the directory or by modifying the entry after it exists to contain the **entryOwner** and **ownerPropagate** information.

It is possible to create an entry owner without specifying the **ownerPropagate** attribute. In this case, the **ownerPropagate** attribute is assumed to have a value of **TRUE** and is added into the directory entry automatically.

Since the **ldapmodify** command is very powerful, all the possible ways of adding the **entryOwner** and **ownerPropagate** information cannot be shown here. The examples shown here describe the more common uses of the **ldapmodify** command to add entry owner information.

Figure 46 on page 285 shows how to add a propagating entry owner with two **entryOwner** values to an existing entry.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f newOwn.ldif
```

Where `newOwn.ldif` contains:

```
dn: cn=tim, o=Your Company
changetype: modify
replace: entryOwner
entryOwner: cn=joe, o=Your Company
entryOwner: cn=carol, o=Your Company
-
```

Figure 46. Example of adding a propagating set of entry owners to existing entry in the directory

The entry owners added in Figure 46 are created as a propagating set of entry owners since the **ownerPropagate** attribute is not specified and so assumed to be **TRUE**. This means that the entry owners will apply to all entries below `cn=tim, o=Your Company` that do not already have an entry owner associated with them.

While it is not required that the LDAP administrator update all entry owner information, the examples in this section all use the administrator as the entry owner updater. Further, the use of `-h 127.0.0.1` implies that the **ldapmodify** commands are performed from the same system on which the LDAP server is running and that the LDAP server is listening on TCP/IP port 389. Refer to the **ldapmodify** command description in *z/OS Integrated Security Services LDAP Client Programming* for more details on the **-h**, **-p**, **-D**, and **-w** command-line options. The entry owner attributes can be updated from any LDAP client as long as the user performing the update has the proper authorization to update the entry owner information.

The entry owner attributes, like the ACL attributes, are defined to be in a special access class called **restricted**. Thus, in order to allow someone other than the LDAP administrator to update the entry owner attributes, they must either be the entry owner or have the proper authorization to **restricted** attributes. See Figure 40 on page 281 for an example of allowing users other than the LDAP administrator the ability to update entry owner information.

Figure 47 shows an example of adding a non-propagating entry owner. A non-propagating entry owner applies only to the entry to which it is attached and not to the subtree of information that might be stored below the entry in the directory.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f newOwn.ldif
```

Where `newOwn.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
replace: entryOwner
entryOwner: cn=george, o=Your Company
entryOwner: cn=jane, o=Your Company
-
replace: ownerPropagate
ownerPropagate: FALSE
-
```

Figure 47. Example of setting up a non-propagating entry owner

Setting up a non-propagating entry owner is similar to setting up a propagating entry owner. The difference is that the **ownerPropagate** attribute value is set to **FALSE**.

Modifying an owner for an entry

Once an entry owner exists for an entry in the directory, it may have to be updated. To do this, the **ldapmodify** command is used. As described earlier in this chapter, while the **ldapmodify** command is used in these examples, what is really being used is an LDAP client application, issuing LDAP modify

operations to the LDAP server. Thus, modifications to entry owner information need not be performed from the same system on which the LDAP server is running.

Modifications to entry owners can be of a number of different types. The most common modifications are to:

- Add an additional **entryOwner** value to the set of entry owners to allow another person or group to control the entry
- Change an entry owner from propagating to non-propagating (not permitted for the GDBM change log suffix, cn=changelog)
- Remove an **entryOwner** value which exists in the entry owner set to disallow another person or group access to control the entry that they had control over before.

Figure 48, Figure 49, and Figure 50 on page 287 show examples of these modifications, respectively.

Figure 48 shows how an additional **entryOwner** value is added to existing entry owner information.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modOwn.ldif
```

Where `modOwn.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
add: entryOwner
entryOwner: cn=george, o=Your Company
-
```

Figure 48. Example of adding an entryOwner attribute value

In Figure 48, `cn=george, o=Your Company` is granted entry owner control of the `cn=jeff, cn=tim, o=Your Company` entry in the directory. The existing entry owner information remains in the entry; the **entryOwner** attribute value for `cn=george, o=Your Company` is added to this information.

Figure 49 shows how to modify an existing entry owner to be non-propagating instead of propagating.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modOwn.ldif
```

Where `modOwn.ldif` contains:

```
dn: cn=tim, o=Your Company
changetype: modify
replace: ownerPropagate
ownerPropagate: FALSE
-
```

Figure 49. Example of modifying the ownerPropagate attribute

In Figure 49, the existing entry owner set for `cn=tim, o=Your Company` is modified to be non-propagating instead of propagating. This means that the entry owner will no longer apply to entries below `cn=tim, o=Your Company` in the directory tree. Instead, the first propagating entry owner set that is found in an entry above `cn=tim, o=Your Company` will be applied to the entries below `cn=tim, o=Your Company`. If no propagating entry owner is found in the entries above `cn=tim, o=Your Company`, then the default entry owner is used.

Figure 50 on page 287 shows how to remove an **entryOwner** value from existing entry owner information:

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modOwn.ldif
```

Where `modOwn.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
delete: entryOwner
entryOwner: cn=george, cn=tim, o=Your Company
-
```

Figure 50. Example of removing a single entryOwner Attribute value

In Figure 50, the **entryOwner** attribute value for `cn=george, cn=tim, o=Your Company` is removed from the entry owner information for entry `cn=jeff, cn=tim, o=Your Company`.

Deleting an owner for an entry

In order to delete an entry owner set that is attached to an entry in the directory, the **entryOwner** and **ownerPropagate** attributes must be deleted from the entry. To do this, use the **ldapmodify** command to delete the entire attribute (all values) from the entry.

Note: This is not allowed for the GDBM change log suffix entry, cn=changelog.

Figure 51 shows an example of deleting an entry owner set from an entry.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f delOwn.ldif
```

Where `delOwn.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
delete: entryOwner
-
delete: ownerPropagate
-
```

Figure 51. Example of deleting an entry owner set from an entry

In Figure 51, the existing entry owner set against `cn=jeff, cn=tim, o=Your Company` is removed. This means that the entry owner information will no longer apply to the entry. Instead, the first propagating entry owner set that is found in an entry above `cn=jeff, cn=tim, o=Your Company` will be applied to `cn=jeff, cn=tim, o=Your Company`. If no propagating entry owner set is found in the entries above `cn=jeff, cn=tim, o=Your Company`, then the default entry owner is used.

Note: The **entryOwner** and **ownerPropagate** attributes must be deleted from the entry even though the **ownerPropagate** attribute might not have been specified during the creation of the entry owner information.

Creating a group for use in ACLs and entry owner settings

Sets of users can be grouped together in the directory by defining them as members of a group in the directory. A directory group, used for access control checking, is just another entry in the directory. A static, dynamic, or nested group entry can be used as a group on the **aclEntry** or **entryOwner** attributes. See Chapter 21, “Static, dynamic, and nested groups,” on page 257 for more information on creating, modifying, and deleting static, dynamic, and nested group entries.

Note: Deleting a user or a group does not have any cascade effect on any ACLs that include that user or group. The user or group DN is not removed from the ACLs. If another user or group is subsequently created with the same DN, that user or group will be granted the privileges of the former user or group.

When defining access controls or entry owner sets, names of group entries can be used in the same place as user entry names. When access control decisions are performed, a user's group memberships are used in determining if a user can perform the action requested.

Groups are added to access control information in just the same way as user entries are added to access control information. Figure 52 shows how a group can be added to the **aclEntry** information in an existing access control specification for an entry. Figure 53 shows how a group can be added as an **entryOwner** to an existing entry owner specification for an entry.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modAcl.ldif
```

Where `modAcl.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
add: aclentry
aclEntry: group:cn=group1, o=Your Company:normal:rwsc:sensitive:rsc
-
```

Figure 52. Example of adding a group to access control information

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modOwn.ldif
```

Where `modOwn.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
add: entryOwner
entryOwner: cn=group1, o=Your Company
-
```

Figure 53. Example of adding a group to entry owner information

Chapter 23. Replication

Once the z/OS LDAP server is installed and configured, users can access the directory, add objects, delete objects, or perform search operations to retrieve particular sets of information.

Replication is a process which keeps multiple databases in sync. Through replication, a change made to one database is propagated to one or more additional databases. In effect, a change to one database shows up on multiple different databases.

There are several benefits realized through replication. The single greatest benefit is providing a means of faster searches. Instead of having all search requests directed at a single server, the search requests can be spread among several different servers. This improves the response time for the request completion.

Additionally, the replica provides a backup to the replicating server. Even if the replicating server crashes, or is unreadable, the replica can still fulfill search requests, and provide access to the data.

There are two types of replication:

- In peer to peer replication, each LDAP peer server is a read-write server. Updates processed on one peer server are replicated to all the other peer servers. Peer servers are read-write to all users.

  Note: The z/OS support for peer to peer replication is provided for failover support purposes. There is no support for resolving conflicting simultaneous updates on multiple peer servers, which can cause a failure of replication. As a result, updates should be targeted to one peer server at a time.
- In read-only replication, a single read-write LDAP server (the master) replicates the updates it processes to a set of read-only replica servers.

  Master
  : All changes to the database are made to the master server. The master server is then responsible for propagating the changes to all other databases. It is important to note that while there can be multiple databases representing the same information, only one of those databases can be the master.

  Read-only replica
  : Each of the additional servers which contain a database replica. These replica databases are identical to the master database. These servers are read-only to all users and will only accept updates from their master server.

A replication network can contain both peer replica servers and read-only replica servers. In this case, each peer server must act as a master to each read-only replica (in addition to being a peer to all the peer servers), so that updates that occur on any peer server are replicated to all the other peer and read-only replicas in the network.

Replication is only supported when the servers involved are running in single-server mode. Although replication is not supported when operating multiple concurrent server instances against the same database (multi-server operating mode), similar benefits are afforded when operating in this mode. Refer to “Determining operational mode” on page 82 for more information about server operating modes.

In z/OS LDAP, replication is only supported in the TDBM (DB2-based) backend.

ibm-entryuuid replication

Replication of the **ibm-entryuuid** will be performed to:

- z/OS V1R3 with the Coexistence PTF applied
- z/OS V1R4 and later releases.

If a replica server is not one of the above and the replica server does not support the **ibm-entryuuid** attribute, then the **ibm-entryuuid** attribute will not be replicated to that server; however, the entry and other attributes will be replicated.

Complex modify DN replication

Replication of Modify DN new superior operations will be performed to z/OS V1R4 and later LDAP replica servers. If a replica server is not at a supported level, the Modify DN new superior operation will fail until the replica is removed from the replica collection or it is upgraded to a LDAP z/OS V1R4 or later server.

Password encryption and replication

To ensure data integrity and the proper working of the LDAP servers in the replication environment, the **pwEncryption** option in the configuration files for the servers involved in replication must be the same. If one of the servers involved in replication is a non-z/OS server, then the administrator must choose a **pwEncryption** method that is supported by both servers for correct operation of replication. If no encryption methods are common between the servers, then password encryption should not be used.

For crypt encryption, note that the values returned by the crypt algorithm are not portable to other X/Open-conformant systems. This means the crypt() algorithm cannot be used for replication between z/OS and a non-z/OS server.

For DES encryption, where both the servers involved in replication are z/OS LDAP servers, the same DES key label and data key must be defined on both z/OS systems through the ICSF KGUP and CKDS facilities. (See the information on managing cryptographic keys in the *z/OS Cryptographic Services ICSF Administrator's Guide* for more details.) This key label must be used in the configuration files of both of the LDAP servers involved in replication.

Replicating server

In order for the replication process to occur, the following must happen:

- The replicating server (master or peer) must be aware of each replica that is to receive the change information.
- Each read-only replica must be aware of the replicating server for the database that it serves. See "LDAP update operations on read-only replicas" on page 294 for more information.

The replicating server becomes aware of the existence of the replica servers when objects (entries) of type **replicaObject** are added to the directory. Each of these objects represents a particular replica server. The attribute/value pairs within the replica object provide the information the replicating server needs in order to find the replica server and send any updates to that server.

Replica objects

The **replicaObject** object class is provided in the system schema file **schema.user.ldif**. Like other LDAP object class definitions, the **replicaObject** has mandatory and optional attributes. Each of the **replicaObject** attributes are single-valued. The following is a description of the mandatory attributes of **replicaObject**.

Table 46. Replica object-schema definition (mandatory attributes)

Attribute	Description and example

replicaHost	This can be an IPv4 address, IPv6 address, or a hostname.
	
	Example:
	`replicahost: 9.130.77.27`
	`replicahost: [5f1b:df00:ce3e:e200:20:800:2078:e3e3]`
	`replicahost: myMachine.ibm.com`

replicaBindDN	Specifies the LDAP distinguished name that the replicating server uses to bind to the replica when sending directory updates. The **replicaBindDN** and the **masterServerDN** or **peerServerDN** in the replica's configuration file must have the same value. Example: `replicaBindDN: cn=Master`
replicaCredentials	Contains the authentication information needed for the replicating server to authenticate to the replica using the **replicaBindDN**. Example: `replicaCredentials: secret`
cn	Forms the RDN of the LDAP distinguished name of the **replicaObject** entry. Example: `cn: myReplica`

In the examples in Table 46, when the replicating server receives and successfully finishes an update request, the update is also sent to `myMachine.ibm.com` on port 389 (the default port). The replicating server performs a bind operation using the DN of `cn=Master` and password of `secret`. See "Establishing the administrator DN and the replica server DN and passwords" on page 87 for more information specifying the replication server DN and password.

In addition, there are several attributes available that provide additional flexibility in configuring a replica server. For instance, an added description could better describe the replica server, and it could listen on a different port then the default port of 389. Examples of adding a description and changing the port to 400 are shown in Table 47, which describes the optional attributes of **replicaObject**.

Table 47. Replica object schema definition (optional attributes)

Attribute	Description and example
replicaPort	Describes the port number on which the replica is listening for incoming requests. By default, the server listens on port 389. Example: `replicaPort: 400`
replicaUpdateTimeInterval	Delays the propagation of additional updates for specified number of seconds. The default is for the replicating server to send updates immediately. Example: `replicaUpdateTimeInterval: 3600`
replicaUseSSL	Determines whether the replicating server should replicate over SSL/TLS. The default is to replicate without using SSL/TLS. Example: `replicaUseSSL: TRUE`

Table 47. Replica object schema definition (optional attributes) (continued)

Attribute	Description and example
description	Provides an additional text field for extra information pertaining to the replica object. Example: `description: Replica machine in the fourth floor lab`
seeAlso	Identifies another directory server entry that may contain information related to this entry. Example: `seeAlso: cn=Alternate Code, ou=Software, o=IBM, c=US`
replicaBindMethod	Identifies the bind method to be used. If it is specified, it must be set to **simple**. Example: `replicaBindMethod: simple`

Replication only supports simple authentication. SASL EXTERNAL, GSSAPI, DIGEST-MD5, and CRAM-MD5 bind mechanisms are not supported as valid replication bind mechanisms.

Localhost suffix

When the LDAP server is configured with TDBM, the **cn=localhost** suffix is no longer required. However, its use is still supported. The **cn=localhost** suffix entry will not be created automatically in TDBM.

Adding replica objects in TDBM

In TDBM, replica objects can be placed anywhere within the directory tree. This also implies that the suffix **cn=localhost** can be removed from the LDAP server configuration file. Placing replica objects in the directory tree then requires that any parent entries of the **replicaObject** entry be added to the directory prior to adding the **replicaObject** entry. These entries must be added to both the replicating server and replica server before addition of the **replicaObject**. This is needed on the replica server because these entries are being added at the replicating server without replication being active. If a replica object is not placed as a leaf node in the directory tree, the only entries allowed below the replica object are other replica objects. The LDAP server will allow non-replica entries to be placed below replica entries; however, these entries will not be replicated to the replica servers. Following is an example of a replica object definition using LDIF format.

```
dn: cn=myReplica,o=YourCompany
objectclass: replicaObject
cn: myReplica
replicaHost: myMachine.ibm.com
replicaBindDn: cn=Master
replicaCredentials: secret
replicaPort: 400
replicaUseSSL: FALSE
description: "Replica machine in the fourth floor lab"
```

Replica server

Initialization, or population, of a replica database requires several steps.

Special note: If the replicating server is configured with TDBM, changes to the schema entry on the replicating server are not replicated. The schema on the replica must be modified by a user bound as the **masterServerDN** or **peerServerDN**. See “Configuring the replica” on page 293 for more information. A separate update of the replica schema will be required each time the schema is updated on the replicating server. Note that if you are modifying the schema on a TDBM read-only replica and are not bound as the

masterServerDN, the **masterServer** configuration option will cause the modification to be redirected to the replicating server. This will cause the schema on the replica and replicating servers to be out of sync. No error message occurs.

Populating a replica

1. Stop the LDAP replicating server.
2. Unload the replicating server's directory contents if there are any entries. For TDBM, use the **tdbm2ldif** utility (see "tdbm2ldif utility" on page 148).
3. Make sure the schema for the replica server is the same as the schema for the replicating server.

   If the replica and replicating server are both z/OS servers configured with TDBM, the schema can be unloaded from the replicating server using **tdbm2ldif** and reloaded into the replica using either the **ldif2tdbm -s** option or **ldapmodify** with the replica server started.
4. Run a load utility with a single added directory entry which defines a **replicaObject** entry into the replicating server's directory contents. For TDBM, use either the **ldif2tdbm** utility (see "ldif2tdbm utility" on page 138) or **ldapadd** with the replicating server running.

   Note that in order to load the **replicaObject** entry, it is also necessary to load any parent entries in the directory hierarchy in hierarchy order.
5. If the replicating server does not contain any entries, no further action must be taken to ensure that the replica and replicating server are in sync and the replicating server can now be restarted; otherwise, continue to the next step.
6. Transport the LDIF file created in step 2 to the replica server's location.
7. Run a load utility on the replica server using the LDIF file from step 6. For TDBM, stop the replica server if it is running and use **ldif2tdbm**.
8. Configure the replica (see next section).
9. Start the replica server. If this is a peer server, ensure that it does not contain a replica object that defines this server as a replica of itself.
10. Start the replicating server.

Configuring the replica

The key to a successful replica configuration rests in ensuring that the values in the **replicaObject** on the replicating server (master or peer) accurately represent the relevant values on the replica server (read-only or peer). Configuring the replica involves specifying appropriate configuration file option values to identify:

- the IP address and port on which the replica server should listen for communication from the replicating server
- the type of connection expected by the replicating server when it communicates to the replica server, either over a non-secure or secure connection
- the DN and password used by the replicating server

Note: **ldif2tdbm** does not replicate changes when adding entries to the replicating server. So, if you are using **ldif2tdbm** to add entries to a replicating server you must also use it to add entries to each replica, with no intervening updates on the replicating server before the replica is loaded.

The following table identifies the relationship between **replicaObject** on a z/OS LDAP replicating server and the configuration options on an IBM replica server. The values specified for these options must be equivalent. An example of what is meant by equivalent is when the replica server is listening on all of its network interfaces, then the **replicaHost** must specify either the corresponding hostname or an IP address of one of the addresses.

Attribute in replica object on replicating server	Corresponding replica server configuration option or command line parameter
replicaHost	The hostname or IP address specified on the **listen** configuration option or the **-l** LDAP server command line parameter.
replicaPort	The port number that is specified on the **listen** configuration option or the **-l** LDAP server command line parameter.
replicaUseSSL	Use of **ldaps://** in the prefix of the **listen** configuration option or the **-l** LDAP server command line parameter corresponds to **TRUE** for **replicaUseSSL**; use of **ldap://** corresponds to **FALSE**.
replicaBindDn	**masterServerDN** or **peerServerDN** configuration option
replicaCredentials	**masterServerPW** or **peerServerPW** configuration option

Notes:

1. **listen** configuration option is only used by the z/OS LDAP server. The value of the **listen** configuration option or **–l** command line parameter is an LDAP URL. For additional information on the **listen** option, see Chapter 8, “Customizing the LDAP server configuration,” on page 53.
2. If the replica server is a non-IBM server, you should consult their documentation for parameters that correspond to the parameters mentioned in the above table.
3. If a read-only replica server is configured with more than one TDBM backend then the **masterServer**, **masterServerDN**, and **masterServerPW** all have to appear in each TDBM backend section in the configuration file and all must have the same values. Similarly, every TDBM section in the configuration file for a peer replica server must have the same values for **peerServerDN** and **peerServerPW**.
4. It is recommended that the **masterServerDN** or **peerServerDN** be a DN that is dedicated specifically to replication. It should not be used for any other operations.
5. The **masterServer**, **masterServerDN**, **masterServerPW**, **peerServerDN**, and **peerServerPW** options must follow the database definition entry in the LDAP server configuration file.
6. Usage of the **masterServerPW** or **peerServerPW** configuration option is strongly discouraged in production environments. See “Establishing the administrator DN and the replica server DN and passwords” on page 87 for alternatives.

LDAP update operations on read-only replicas

Update operations, such as add, delete, modify, and rename, should not be performed against a read-only replica server. Changes must be made to the master server, which then propagates the change to the read-only replica.

If update operations are sent to a read-only replica server, the **masterServer** set in the read-only replica configuration is returned and the operation is referred to the master server and is then propagated to the read-only replica server. For compatibility purposes, if **masterServer** is not specified, the first default referral found in the read-only replica configuration file is used as the master server and all update operations are referred to that server.

In order to maintain database integrity, a load utility (**ldif2tdbm**) should be used on a read-only or peer replica only when initially populating the replica database. If a load utility is used to add entries to a replica server after initial population, these changes are not reflected in the master database. The replica database is corrupted and could give erroneous information.

See “SSL/TLS and replication” on page 297 for information about securing a database.

Changing a read-only replica to a master

When using read-only replication, it may become desirable to change one of the read-only replicas to be the master. Perhaps the machine where the replica server is installed is being upgraded, and the customer wishes this replica to now be the master LDAP server.

The following procedure should be followed to change a read-only replica to a master:

1. Ensure all of the data (including **replicaObject** entries for each replica object) from the master resides in the replica. This can be done by dumping both databases using an unload utility and comparing the output. (For TDBM, use **tdbm2ldif**.) If the replica is out of sync with the master, follow the procedure for correcting out-of-sync conditions.
2. Remove the **replicaObject** entry for this replica from the master server.
3. Stop the master and the replica servers.
4. Remove the **masterServer**, **masterServerDN**, and **masterServerPW** directives from the replica's configuration file.
5. If the original master is being eliminated, simply drop the databases on the original master. (See "Creating the DB2 database and table spaces for TDBM or GDBM" on page 42 for examples of the SPUFI commands needed to drop the databases.) The new master can now be started.
6. If the original master is going to become a replica:
   a. Add a **replicaObject** for the original master to the new master database using **ldapadd** or a load utility (**ldif2tdbm** for TDBM).
   b. Add the **masterServer** directive to the new replica's configuration file. Make sure it points to the new master.
   c. Add the **masterServerDN** and **masterServerPW** directives to the new replica's configuration file.
   d. Start the servers.

Peer to peer replication

z/OS LDAP peer replication server provides failover support. With this support, if a LDAP server fails, the peer replication server can take over the role of the failing LDAP server and it is then available to process LDAP operations.

A z/OS LDAP peer replication server is a read/write replication server that can send and receive replicated entries. An LDAP server can have both peer replication servers and read-only replication servers defined as **replicaObject** entries.

Note: Peer to peer replication uses the same replica objects and attribute values as shown in Table 46 on page 291. The instructions in "Adding replica objects in TDBM" on page 292 also apply to peer replicas.

A peer to peer replication environment can be as simple as two LDAP servers that are peers to each other, or as complicated as several LDAP servers, where some servers are read-only replication servers and the other servers are peer replication servers. Every peer replication server must replicate to all other peer and read-only replication servers.

Server configuration

Two new configuration options, **peerServerDn** and **peerServerPW**, have been added to support peer replication. See Chapter 8, "Customizing the LDAP server configuration," on page 53 for more information.

Note: Usage of the **peerServerPW** configuration option is strongly discouraged in production environments. See "Establishing the administrator DN and the replica server DN and passwords" on page 87 for alternatives.

Maintenance mode

Maintenance mode is the LDAP server setup mode for replication. This mode allows the LDAP server to be primed for replication. While in maintenance mode, the LDAP server only allows operations by either the **masterServerDN** or the **peerServerDN**. Pending replication entries are replicated to the other replica servers.

Note: The **adminDN** is not allowed access to the LDAP server in maintenance mode.

Specify the **-m** option on the server startup command to start the LDAP server in maintenance mode.

The operator **MODIFY** command can be used to change from maintenance mode to normal mode while the LDAP server is running. It can also be used to put a running server into maintenance mode. The following command can be sent to the LDAP server from the SDSF or the operator's console. If the command is entered from SDSF, it must be preceded by a slash (/).

```
f ldapsrv,appl=maintmode=state
```

where *state* can be **on** to turn maintenance mode on or **off** to turn maintenance mode off (and turn normal mode on).

Conflict resolution

Minimal conflict resolution is done in a peer environment. For example, if peer replication server A receives an update to entry E at the same moment that peer B receives a delete of the same entry, replication can stall on server A. To avoid this, ensure that your peer servers are not receiving conflicting operations.

When a conflict occurs, a notification will be sent to the console and server log.

Adding a peer replica to an existing server

For failover support, it may be necessary for you to add a peer replica to an existing server or set of servers. These servers can be stand-alone or already actively replicating.

In order to add a peer replica to a z/OS LDAP server, you should do the following:

1. Start the new peer replica in maintenance mode. The peer replica must have a **peerServerDN** and **peerServerPW** defined in the configuration file.
2. Stop the existing servers. For each existing server that is not already a peer server, update its configuration file to include the **peerServerDN** and **peerServerPW** configuration options. Restart the existing read-write servers in maintenance mode.
3. Prime the new peer replica with all the data from an existing server. You can accomplish this by dumping the existing server's database (for TDBM use **tdbm2ldif**) and adding the data to the new peer replica (for TDBM use **ldif2tdbm**). Refer to "Populating a replica" on page 293 for more information.
4. Add a replica object to the existing servers to point to the new peer replica.
5. Add a replica object in the new peer replica pointing to the existing server that was used to prime this server.

   Note: If the existing server was a replicating server with replica objects defined to it, those replica objects would have been copied to the new peer replica in step 3 above. Ensure that this server does not contain a replica object that defines this server as a replica of itself.
6. Turn off maintenance mode on all servers.

The existing servers and the new peer replica are now peer read-write replicas.

Upgrading a read-only replica to be a peer replica of the master server

It may be necessary for you to upgrade a read-only replica to a peer of its master, for example, if a peer of the master failed or further failover support is needed.

You should do the following to change a read-only replica to a peer replica:

1. Stop both the master server and the read-only replica.
2. Remove the **masterServer** options from the configuration file of the read-only replica.
3. Add a **peerServerDN** and **peerServerPW** to each server's configuration file. The two servers will now be peer servers.
4. Start both servers in maintenance mode.
5. On the read-only replica being upgraded:
   - Add a replica object for each replica that the master server points to (except the object that previously pointed to the read-only replica that is being upgraded). This can include both peer servers and read-only replicas. Note that the master server might have other peer servers.
   - Add a replica object to point to the master.
6. On the master, ensure that the credentials are valid in the **replicaObject** for the read-only replica being upgraded.
7. Turn off maintenance mode on both servers.

The read-only replica and the master server are now peer read-write replicas.

Downgrading a peer server to read-only replica

It may be necessary for you to downgrade a peer server to a read-only replica, for example, if a previously upgraded read-only replica is no longer required to be a peer server, or to prevent out-of-sync conditions between peer servers.

You should do the following to downgrade a peer server to a read-only replica:

1. Stop the peer server.
2. Remove the **peerServerDN** and **peerServerPW** options from the configuration file.
3. Add **masterServer**, **masterServerDN**, and **masterServerPW** options to the peer replica, choosing one server (it must be a peer replica or a master server) to act as its **masterServer**.
4. Ensure that the credentials are valid in the **replicaObject** for the newly downgraded peer server on all the replicating servers.
5. Start the server.

The peer server is now a read-only replica.

SSL/TLS and replication

SSL/TLS can be used to communicate between a replicating server (master or peer) and a replica server (read-only or peer).

Replica server with SSL/TLS enablement

Set the replica server up for SSL/TLS just like a normal SSL/TLS server. It needs its own public-private key pair and certificate, and the configuration file needs the standard SSL keywords set (**sslKeyRingFile** and **sslKeyRingFilePW** configuration file options). See "Setting up for SSL/TLS" on page 46 for more information.

Replicating server with SSL/TLS enablement

The replicating server acts like an SSL/TLS client to the replica server.

To set up the replicating server, you must:

1. Run the **gskkyman** utility (see *z/OS Cryptographic Services System Secure Sockets Layer Programming*), this time as if you were a client. You should use the same key database file that contains the replicating server's key pair and certificate. Receive the replica's self-signed certificate and mark it as trusted.
2. In the replicating server's configuration file:
   - Set the **sslKeyRingFile** to the replica key database file name created above.
   - Set the **sslKeyRingFilePW** to the password for the replica key database file, or set **sslKeyRingPWStashFile** to the file name where the password is stashed.
3. In the replica object:
   - Set the **replicaPort** keyword to the replica's secure port number.
   - Set the **replicaUseSSL** keyword to **TRUE**.

See "Setting up for SSL/TLS" on page 46 for more information.

Since the replicating server acts like an SSL/TLS client to the replica server, the replicating server binds with the replica server. The bind method used is **simple** bind. The SASL external bind method is not supported for replication.

Troubleshooting

If the replica server does not seem to be receiving updates from the replicating server (master or peer), there are several possible reasons. Check the following conditions for a possible quick fix:

- Check for messages from the replicating server.
- Verify that the replica object exists in the replicating database, and was specified correctly to match with the replica server. If **cn=localhost** is used as the suffix for all replica objects, perform an **ldapsearch** with a base of **cn=localhost** and a filter of **objectClass=***. Otherwise, perform an **ldapsearch** where the search base is the suffix defined in the configuration file and the filter is **objectClass=replicaObject**. If more than one suffix is configured for TDBM, the search must be repeated using each suffix in the search base.

  See *z/OS Integrated Security Services LDAP Client Programming* for more information about **ldapsearch**.
- Verify that the **replicaHost** value in the replica object for that replica specifies the machine on which the replica is running.
- Check that the values listed in the replica object for that replica match those of the replica server configuration. Specifically, the **replicaPort**, **replicaBindDN**, and **replicaCredentials** should be verified.
- Check that the **replicaUpdateTimeInterval** specified in the replica object for that replica has been set correctly.
- Verify that the replica server is running by performing an **ldapsearch** against the replica.
- Check that the default referral specified in the replica's configuration file points to the replicating server.
- If the **replicaObject** attribute **replicaUseSSL** is set to **TRUE**, verify the **replicaObject** attribute **replicaPort** is set to the SSL port configured on the replica server, and verify the **sslKeyRingFile**, and **sslKeyRingFilePW** or **sslKeyRingPWStashFile** values in the replicating's configuration file are correct.
- When adding a large number of entries, ensure that the region size for the replicating server is sufficient for replicating the entries to the replica. Entries on the replicating server are kept in memory during replication. If the region size is not sufficient, an out of memory condition can occur in the LDAP server. If possible, set the region size on the replicating server to 0M (or unlimited). If that cannot be done, set the region size to 14M (needed to run the LDAP server itself) plus two times the size of the largest LDIF file that is to be added to the replicating server.

Recovering from out-of-sync conditions

If a replica becomes out-of-sync with its replicating server for any reason, and normal replication processing is not correcting the situation, it may be necessary to reload the replica.

The following procedure should be followed to reload a replica:

1. Delete the **replicaObject** entry or entries on the replicating server. This resets the replication information in the replicating server. It is desirable to search these entries first, obtaining LDIF format output for these entries. This LDIF can be used to reload the **replicaObjects** in step 7.
2. Stop the replicating server and all the replica servers.
3. Drop and recreate the table spaces on the replica servers. (See “Creating the DB2 database and table spaces for TDBM or GDBM” on page 42 or for an example of the SPUFI commands needed to drop and recreate the table spaces.)
4. Run an unload utility on the replicating server. For TDBM, use **tdbm2ldif** twice, once to unload the schema and a second time to unload the database entries.
5. Run a load utility on each replica, using the data retrieved from the replicating server, above. For TDBM, use **ldif2tdbm** (both the schema and the database entries can be loaded in one invocation).
6. Start the replicating server and replica servers. Make sure there is no update activity on the replicating server.
7. Add the **replicaObject** entry or entries back to the replicating server. The LDIF output captured from the search in step 1 can be used to add the entries back into the replicating server.

Chapter 24. Alias

Alias support provides a means for a TDBM directory entry to point to another entry in the same TDBM directory. An alias can also be used to create a convenient public name for an entry or subtree, hiding the more complex actual name of the entry or subtree. Aliasing is only supported in the TDBM (DB2 based) backend.

Alias support involves:

- Creating an alias entry which points to another entry
- Dereferencing during search: when a distinguished name contains an alias, the alias is replaced by the value it points to and search continues using the new distinguished name.

For example, you can create an alias entry to provide a simple name for the ZOSLDAP department:

```
"o=LDAPZOS,o=IBM"
```

The alias entry points to the actual ZOSLDAP department:

```
"ou=DEPTC8NG, o=Poughkeepsie,o=IBM_US,o=IBM"
```

This provides easier access to the entries of the ZOSLDAP developers, using public names such as:

```
"cn=kmorg,o=LDAPZOS,o=IBM"
```

This name is dereferenced during search to:

```
"cn=kmorg,ou=DeptC8NG,ou=Poughkeepsie,o=IBM_US,o=IBM
```

and the information for that entry is returned.

Impact of aliasing on search performance

Usage of aliases in a directory can cause a large increase in the amount of processing that takes place during search, even if no alias entries are actually involved in the particular search that was requested. To minimize the impact to search performance:

- Do not add aliases to the directory if they are not needed. There is no impact on search if there are no aliases in the directory.
- Only perform a search with dereferencing when aliases are involved in the search. Again, the impact on search is avoided if no dereferencing is requested.

  Note: The search request from the LDAP client specifies whether to do dereferencing. The default value for dereferencing varies between different LDAP clients. If the default is to do dereferencing (this is the case with some Java™ clients), make sure to specifically reset this value to do no dereferencing when you issue search requests for which you do not want to do dereferencing.
- If you do want to use aliases in a directory, use them efficiently to minimize the number of alias entries. For example, use an alias entry for the root of a subtree (such as the alias for a department entry in the example above) rather than creating an alias entry for each individual entry within the subtree.

Alias entry

An alias entry contains:

- one of two object classes:
  - **aliasObject** - AUXILIARY object class
  - **alias** - STRUCTURAL object class.

Note: This requires an object class such as **extensibleObject** to allow the naming attributes for the entry.

- **aliasedObjectName** attribute
  - its value is the distinguished name that the alias points to

These object classes and attributes are part of the TDBM schema when a TDBM backend is started.

Below is an example of an alias entry:

```
dn: ou=LDAPZOS,o=IBM
objectclass: organizationalUnit
objectclass: aliasObject
ou: LDAPZOS
aliasedobjectname: ou=DeptC8NG,ou=Poughkeepsie,o=IBM_US,o=IBM
```

or

```
dn: ou=LDAPZOS,o=IBM
objectclass: alias
objectclass: extensibleobject
ou: LDAPZOS
aliasedobjectname: ou=DeptC8NG,ou=Poughkeepsie,o=IBM_US,o=IBM
```

Alias entry rules

1. An alias entry must be a leaf entry; thus, an alias entry cannot have any descendant entries. This also means that no ancestor of an entry can be an alias entry.
2. A suffix entry can be an alias entry. In this case, the suffix entry will have no entries below it.
3. The schema entry cannot also be an alias entry. A referral entry cannot also be an alias entry.
4. An alias entry cannot point to itself or to below itself. Also, an alias entry cannot point to the schema entry. These checks are performed when an alias entry is created, modified, or renamed.

Dereferencing

All or part of a distinguished name (DN) can be an alias. Dereferencing a DN consists of the systematic replacement of an alias within the DN by the value of the **aliasedObjectName** attribute of the alias entry. This creates a new DN that must then be checked to see if it contains an alias that needs to be dereferenced. This continues until the resulting DN contains no aliases (or a loop is detected).

The final dereferenced DN contains must be the DN of an existing entry in the same backend as the original DN. The final dereferenced DN cannot be a referral or be under a referral (even if the **manageDsaIT** control is set to process a referral as a normal entry). These checks are performed when an alias is used during search.

Loop detection is performed during dereferencing and an error is returned if the same alias entry is encountered more than once during dereferencing. Also, there is a limit to the number of relative distinguished names (RDNs) that can be involved in dereferencing. The limit is 256 RDNs.

Alias dereferencing can only be performed during search operations. In all other operations (such as bind, add, modify, delete, compare, and rename), an alias entry is always processed like a 'normal' entry, without dereferencing.

No access control (ACL) permissions are checked when dereferencing an alias. Normal access control is performed against the dereferenced entries that match the search filter. Thus a search can dereference aliases even though the requestor might not have any permissions to those alias entries.

Dereferencing during search

Dereference options

A flag value controls what alias dereferencing will be done during a search operation. This flag is sent by the client on the search request. The flag can have one of four values:

LDAP_DEREF_NEVER (0)
do not dereference any alias entries. Alias entries encountered during the search operation are processed as 'normal' entries and are returned if they match the search filter.

LDAP_DEREF_SEARCHING (1)
dereference alias entries within the scope of the search but do not dereference the search base entry (if it contains an alias). The search base is processed as a 'normal' entry (even if it is an alias entry) and is returned if it matches the search filter and is in the search scope.

LDAP_DEREF_FINDING (2)
dereference the search base entry (if it contains an alias) but do not dereference any other alias entries within the search scope. Alias entries within the search scope of the derefereneced base are processed as 'normal' entries and are returned if they match the search filter.

LDAP_DEREF_ALWAYS (3)
dereference the search base entry (if it contains an alias) and dereference alias entries within the scope of the search. All alias entries encountered during the search operation are dereferenced.

Dereferencing during finding the search base

In a search request with **LDAP_DEREF_FINDING** or **LDAP_DEREF_ALWAYS**, dereferencing the search base just establishes a new search base. The results are equivalent to those from a search request that specifies the new base is its base.

Dereferencing during searching in subtree searches

In a search request with **LDAP_DEREF_SEARCHING** or **LDAP_DEREF_ALWAYS** and subtree scope, dereferencing each entry under the base produces additional bases of subtrees to be searched. The aliases under each additional base are also dereferenced during search to find yet more subtree bases, and so on. When all the additional subtrees have been identified, the search filter is applied to all the non-alias entries in all the subtrees and the entries that match the filter are returned.

Dereferencing during searching in one-level searches

In a search request with **LDAP_DEREF_SEARCHING** or **LDAP_DEREF_ALWAYS** and one-level scope, dereferencing each alias entry that is one level below the search base yields additional entries to search (even though they are no longer one level below the search base). The search filter is then applied to these additional entries and to the non-alias entries that are one level below the search base and the entries that match the filter are returned.

In all cases the same entry will not be returned more than once, even if it appears multiple times within the search scope due to alias dereferencing.

Dereferencing and referrals

When dereferencing a DN, the resulting DN cannot be the DN of a referral entry or under a referral entry. This results in a dereferencing error. The resulting DN can be above a referral entry; in this case, regular referral processing is performed.

Errors during dereferencing

The common dereferencing errors and the resulting return codes are:

- loop detected during dereferencing: LDAP_ALIAS_PROBLEM (x'21')
- no entry in this backend for dereferenced DN: LDAP_ALIAS_DEREF_PROBLEM (x'24')
- dereferencing to or under a referral: LDAP_UNWILLING_TO_PERFORM (x'35')

Alias examples

The following figure show the directory structure used in the examples. The dashed lines indicate aliases.

Note: Fictitious attributetypes are used in the figure.

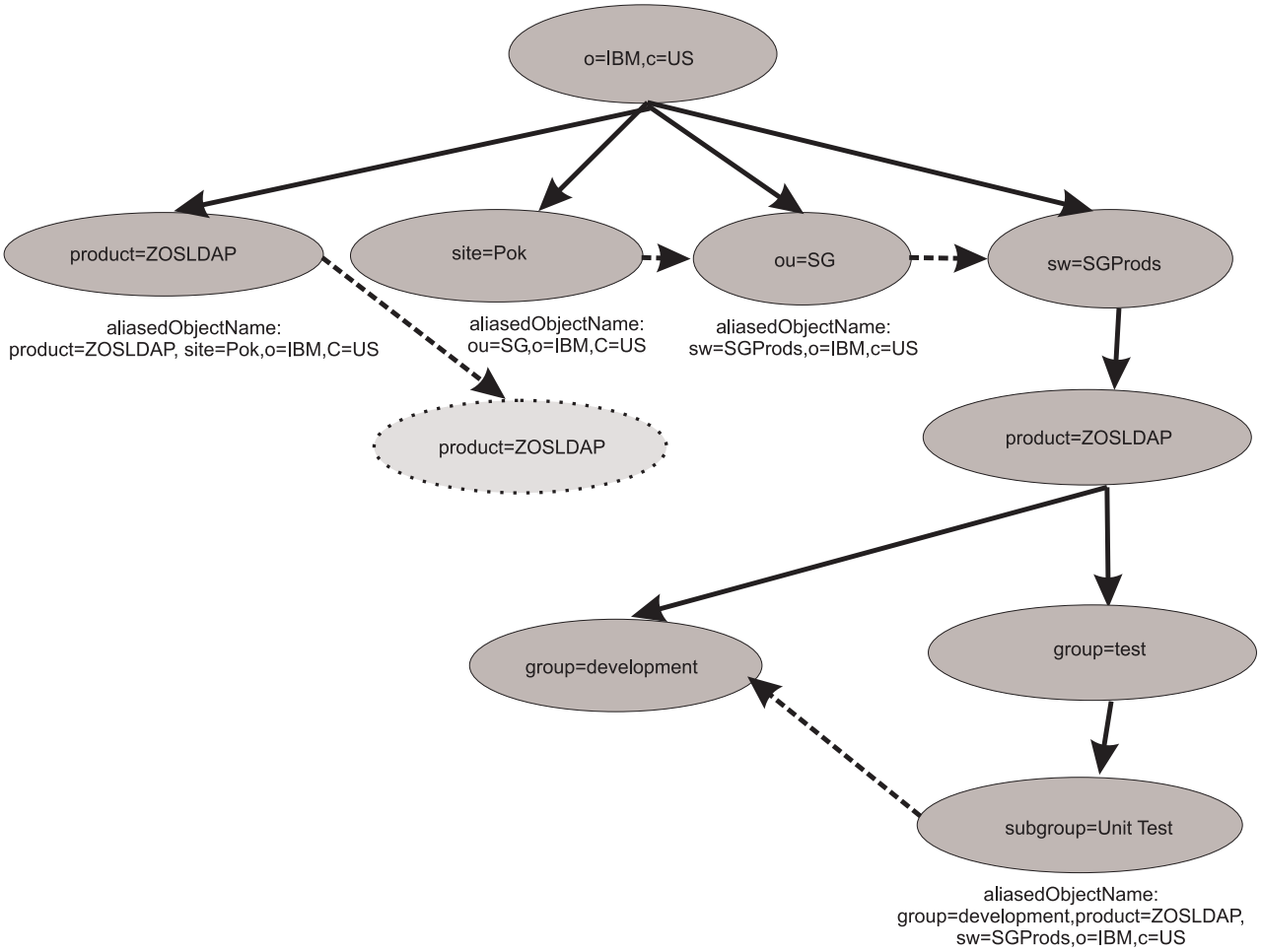


Figure 54. Alias example

The following search examples show the entries that are returned for various combinations of search base, search scope, and dereference option. The filter in each example is ″objectclass=*″. Cases that are affected by alias dereferencing are indicated with an ″*″.

Example #1: Perform a search from the base ″sw=SGProds, o=IBM, c=US″.

scope = base

- Returned entries with LDAP_DEREF_NEVER, LDAP_DEREF_SEARCHING, LDAP_DEREF_FINDING, or LDAP_DEREF_ALWAYS specified:

  ```
  "sw=SGProds, o=IBM, c=US"
  ```

scope = one-level

- Returned entries with LDAP_DEREF_NEVER, LDAP_DEREF_SEARCHING, LDAP_DEREF_FINDING, or LDAP_DEREF_ALWAYS specified:

  ```
  "product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
  ```

scope = subtree

- Returned entries with LDAP_DEREF_NEVER or LDAP_DEREF_FINDING specified:

```
1. "sw=SGProds, o=IBM, c=US"
2. "product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
3. "group=test, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
4. "group=development, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
5. "subgroup=Unit Test, group=test, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
```

- * Returned entries with LDAP_DEREF_SEARCHING or LDAP_DEREF_ALWAYS specified

```
1. "sw=SGProds, o=IBM, c=US"
2. "product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
3. "group=test, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
4. "group=development, product=ZOSLDAP, sw=SGProds, o=IBM, c=US" (returned only once)
```

Example #2: Perform a search from the base ″site=Pok, o=IBM, c=US″.

scope = base

- Returned entries with LDAP_DEREF_NEVER or LDAP_DEREF_SEARCHING specified:

```
"site=Pok, o=IBM, c=US"
```

- * Returned entries with LDAP_DEREF_FINDING or LDAP_DEREF_ALWAYS specified:

```
"sw=SGProds, o=IBM, c=US"
```

scope = one-level

- Returned entries with LDAP_DEREF_NEVER or LDAP_DEREF_SEARCHING specified:

```
No entries returned
```

- * Returned entries with LDAP_DEREF_FINDING or LDAP_DEREF_ALWAYS specified:

```
"product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
```

scope = subtree

- Returned entries with LDAP_DEREF_NEVER or LDAP_DEREF_SEARCHING specified:

```
"site=Pok, o=IBM, c=US"
```

- * Returned entries with LDAP_DEREF_FINDING specified:

```
1. "sw=SGProds, o=IBM, c=US"
2. "product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
3. "group=test, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
4. "group=development, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
5. "subgroup=Unit Test, group=test, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
```

- * Returned entries with LDAP_DEREF_ALWAYS specified:

```
1. "sw=SGProds, o=IBM, c=US"
2. "product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
3. "group=test, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
4. "group=development, product=ZOSLDAP, sw=SGProds, o=IBM, c=US" (returned only once)
```

Example #3: Perform a search from the base ″product=ZOSLDAP, o=IBM, c=US″.

scope = base

- Returned entries with LDAP_DEREF_NEVER or LDAP_DEREF_SEARCHING specified:

```
"product=ZOSLDAP, o=IBM, c=US"
```

- * Returned entries with LDAP_DEREF_FINDING or LDAP_DEREF_ALWAYS specified:

```
"product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
```

scope = one-level

- Returned entries with LDAP_DEREF_NEVER or LDAP_DEREF_SEARCHING specified:

```
No entries returned
```

- * Returned entries with LDAP_DEREF_FINDING or LDAP_DEREF_ALWAYS specified:

```
1. "group=test, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
2. "group=development, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
```

scope = subtree

- Returned entries with LDAP_DEREF_NEVER or LDAP_DEREF_SEARCHING specified:

```
"product=ZOSLDAP, o=IBM, c=US"
```

- * Returned entries with LDAP_DEREF_FINDING specified:

  1. "product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
  2. "group=test, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
  3. "group=development, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
  4. "subgroup=Unit Test, group=test, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"

- * Returned entries with LDAP_DEREF_ALWAYS specified:

  1. "product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
  2. "group=test, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
  3. "group=development, product=ZOSLDAP, sw=SGProds, o=IBM, c=US" (returned only once)

Example #4: Perform a search from the base ″group=test, product=ZOSLDAP, o=IBM, c=US″.

scope = base

- Returned entries with LDAP_DEREF_NEVER or LDAP_DEREF_SEARCHING specified:

  Error - LDAP_NO_SUCH_OBJECT

- * Returned entries with LDAP_DEREF_FINDING or LDAP_DEREF_ALWAYS specified:

  "group=test, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"

scope = one-level

- Returned entries with LDAP_DEREF_NEVER or LDAP_DEREF_SEARCHING specified:

  Error - LDAP_NO_SUCH_OBJECT

- * Returned entries with LDAP_DEREF_FINDING specified:

  "subgroup=Unit Test, group=test, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"

- * Returned entries with LDAP_DEREF_ALWAYS specified:

  "group=development, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"

scope = subtree

- Returned entries with LDAP_DEREF_NEVER or LDAP_DEREF_SEARCHING specified:

  Error - LDAP_NO_SUCH_OBJECT

- * Returned entries with LDAP_DEREF_FINDING specified:

  1. "group=test, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
  2. "subgroup=Unit Test, group=test, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"

- * Returned entries with LDAP_DEREF_ALWAYS specified:

  1. "group=test, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"
  2. "group=development, product=ZOSLDAP, sw=SGProds, o=IBM, c=US"

Chapter 25. Change logging

The change log is a set of entries in the directory that contain information about changes to objects. Depending on configuration options, information about a change to a TDBM entry or to an object controlled by an application (for example, a RACF user profile) can be saved in a change log entry. An LDAP search operation can be used to retrieve change log entries to obtain information about what changes have taken place.

Each LDAP server contains one change log. The change log entries are created in the same order as the changes are made and each change log entry is identified by a change number value, beginning with 1, that is incremented each time a change number is assigned to a change log entry. Thus, the change number of a new change log entry is always greater than all the change numbers in the existing change log entries.

The change log is implemented in a new type of backend, GDBM. The GDBM backend is envisioned as the LDAP server's global backend. The change log uses a hard-coded suffix, cn=changelog. This suffix is a semi-reserved name: when the GDBM backend is configured, the change log root (cn=changelog) must not overlap any suffix in any TDBM or SDBM backend, and the change log suffix cannot be the source or target of a rename operation. If GDBM is not configured, the user can use cn=changelog as a 'normal' suffix in a TDBM or SDBM backend, however, we do not recommend this because that suffix will have to be renamed to avoid an overlap if GDBM is configured in the future.

Change logging is enabled by configuring GDBM in the configuration file. Change log processing is controlled by additional new configuration options in the GDBM backend. The **changeLogging** configuration option turns change logging on/off. The **changeLogMaxEntries** and **changeLogMaxAge** configuration options determine when removal of old change log entries takes place. See Chapter 8, "Customizing the LDAP server configuration," on page 53 for more information. If none of these configuration options is specified in the GDBM section, the default is to start change logging with no limits on the size of the change log.

If the GDBM backend is configured and the cn=changelog root entry does not exist in the GDBM backend when the server is started, the LDAP Server generates the root entry. The root entry is created with an ACL that allows only the administrator to access the change log. The ACL is propagated to the change log entries. The user needs to use an LDAP modify operation to change this ACL to an appropriate ACL for his usage of the change log.

When the LDAP server is started and change logging is enabled, an informational message is issued to indicate change logging is enabled, on/off, change log limits, and the number and range of change log entries. An error message is issued to indicate that changes will not be logged if the GDBM backend configuration fails.

Configuring the GDBM backend

Note: You can use the LDAP configuration utility, ldapcnf, to configure GDBM.

The following configuration file options are required to configure GDBM:

```
database GDBM GLDBGDBM [name]
dbuserid dbowner
servername string
```

The existing **attrOverflowSize, dsnaoini, include, multiserver, readOnly, sizeLimit,** and **timeLimit** configuration options can also be specified in the GDBM configuration section. The new **changeLogging,**

changeLoggingParticipant, changeLogMaxAge, and **changeLogMaxEntries** configuration options can also be specified in the GDBM backend. See Chapter 8, “Customizing the LDAP server configuration,” on page 53.

Note: The **suffix** option is not allowed in the GDBM backend.

You can configure a maximum of one GDBM backend in the configuration file.

The GDBM backend uses DB2 to store its entries. The GDBM database is identical to the current TDBM database and is created in the same way using the same SPUFI scripts. A GDBM backend cannot share a database with a TDBM backend.

When the LDAP server is started with GDBM configured, the GDBM schema contains all the object classes and attributes used by the change log root entry and the change log entries.

Additional required configuration

Additional configuration for RACF to be able to log changes to a RACF user:

- The SDBM backend must be configured. The SDBM suffix is needed to create a DN for the change log entry for a modification to a RACF user. SDBM is also needed to retrieve the RACF user's new password or other changed fields.
- LDAP Program Callable support must be enabled in the LDAP Server containing the change log. To do this, add the following option to either the global section of the configuration file or to the command used to start the LDAP Server:

  ```
  listen ldap://:pc
  ```

There is no additional configuration needed to log changes to a TDBM or GDBM entry. If you do not want to create change log entries for changes to entries within a TDBM or GDBM backend, add the following configuration option to that backend section:

```
changeLoggingParticipant no
```

When changes are logged

RACF changes

A new extended operation, **changeLogAddEntryRequest**, is provided to allow an application to log changes to data that it controls. The initial use of this interface is by RACF to log changes to a RACF user, when a user profile is added, modified, or deleted. The RACF changes can be driven through the z/OS LDAP server or be made directly to RACF. For a user password change, RACF intends to include information that the password changed in the change log entry. For other user changes, RACF does not plan to provide specific field information at this time.

The creation of a change log entry when using this interface is entirely separate from the change to RACF, even if the RACF change is made using z/OS LDAP. The result is that a RACF change can occur without a change log entry being created (for example, if the LDAP server is not running or if the change log entry creation fails).

TDBM and GDBM changes

If change logging is activated, each add, modify, delete, or modrdn operation to an entry, including schema entries and the change log root entry, in any TDBM or GDBM backend in the server results in the creation of a change log entry, with the exception of the following:

- Changes made by using **ldif2tdbm** (bulkload) utility are not logged. This includes modifications of the schema, as well as adding new entries.
- A delete or modify operation of a change log entry is not logged.

- If the **changeLoggingParticipant no** configuration option is specified for this backend, then no changes in this backend are logged.

The creation of the change log entry is performed in the same unit of work as the underlying change to TDBM or GDBM that generates the change log entry. This means that the TDBM/GDBM change and the creation of the change log entry are committed at the same time, either both succeed or both are rolled back.

Change log schema

The following object classes and their attributes define a change log entry.

- objectclass: **changeLogEntry**

  changenumber
  an integer assigned to this change log entry

  targetDN
  the DN to which the change was applied. For RACF, this DN is created from a userid passed in by RACF and the SDBM suffix.

  changeType
  add | **modify** | **delete** | **modrdn**

  changeTime
  the timestamp of when the change is made (not when this entry is created)

  changes
  the added entry or the modifications, in LDIF format. is fully supported for change log entries created by TDBM and GDBM. For change log entries created by RACF, this attribute is only present when a RACF user password is changed, and will contain:

  ```
  replace: racfPassword
  racfPassword: *ComeAndGetIt*
  -
  ```

  newRDN
  the new RDN specified in a TDBM modrdn operation

  deleteOldRdn
  a boolean indicating if the old RDN was deleted in a TDBM modrdn operation

  newSuperior
  the new superior distinguished name specified in a TDBM modrdn operation

- objectclass: **ibm-changelog**

  ibm-changeInitiatorsName
  the DN of the entity that initiated the change. For RACF, this DN is created from a userid passed in by RACF and the SDBM suffix.

  Note: If a RACF user's password is changed using the *oldpassword/newpassword* support on a bind to the SDBM backend or on a bind using native authentication, the **ibm-changeInitiatorsName** value is created from the userid under which the LDAP server is running (and not the bound user).

The object classes and attributes used by the change log are contained in the GDBM internal schema. No schema needs to be added to GDBM to do change logging.

The change log root entry and change log entries also have the standard operational attributes: the ACL attributes, **creatorsname, createtimestamp, modifiersname, modifytimestamp**, and **ibm-entryuuid** (change log root only).

Change log entries

The change log consists of:

- One root (suffix) entry, named cn=changelog
- One change log schema entry, named cn=schema,cn=changelog
- One or more leaf entries, named changenumber=*nnn*,cn=changelog

root entry

The change log root entry is generated by the LDAP server, when change logging is first enabled. The root entry cannot be created, renamed, or deleted by the user. The generated root entry contains a propagated ACL that allows only the administrator to access the change log. An appropriately authorized user can modify the root entry to change the ACL. A modification of the change log root results in the creation of a change log option if change logging is on and GDBM is participating in change logging. Operations on the change log root are not replicated.

The generated root entry is:

```
dn: cn=changelog
objectclass: container
cn: changelog
aclentry: group:cn=Anybody
aclPropagate: TRUE
entryowner: access-id:adminDn
ownerProgagate: TRUE
```

The change log root entry should be modified using the modify operation to set access control for the change log. The root entry ACL is always propagated to provide access control to the change log entries because change log entries are not created with their own ACL. The change log root entry can be modified as long as change logging is enabled (the GDBM backend is configured), even if change logging is not on.

leaf entry

Each change log entry is created as a leaf entry directly under the change log root entry, using the **changeLogEntry** and **ibm-changelog** objectclasses and attributes as described above.

- When created, change log entries do not have an ACL specified, thus inherit the ACL propagated from the change log root entry. An ACL can be added to a change log entry using a modify operation.
- Change log entries are only created by the LDAP server. The user cannot directly add a change log entry. Also, the user cannot rename a change log entry.
- A change log entry can be modified, mainly to set an ACL. Most of the attributes in the change log entry are not modifiable by the user.
- Change log entries are deleted by the LDAP server when the change log is trimmed due to reaching a limit specified by the **changeLogMaxEntries** and **changeLogMaxAge** options in the configuration file. Change log entries can also be deleted by the user through a normal delete operation.
- User operations (search, modify, delete) on change log entries are allowed as long as change logging is enabled (the GDBM backend is configured), even if change logging is off. Add and trim operations by the LDAP server are not performed when change logging is off.
- Operations on change log entries are not replicated and do not result in the creation of change log entries.
- The information in the **changes** attribute is complete for change log entries created as a result of changes to TDBM and GDBM entries. For change log entries created by applications, the changes attribute contains whatever values are passed by the application. See the *z/OS Security Server RACF Security Administrator's Guide* for more information on the changes attribute.

The following is an example of a change log entry created by RACF:

```
dn: CHANGENUMBER=1815,CN=CHANGELOG
objectclass: CHANGELOGENTRY
objectclass: IBM-CHANGELOG
objectclass: TOP
changenumber: 1815
targetdn: RACFID=KEN,PROFILETYPE=USER,CN=MYRACF
changetime: 20030611161820.374472Z
changetype: MODIFY
changes: replace: racfPassword
racfPassword: *ComeAndGetIt*
-

ibm-changeinitiatorsname: RACFID=SUADMIN,PROFILETYPE=USER,CN=MYRACF
```

Searching the change log

The change log can be searched using the standard LDAP search APIs or command utilities.

- You can use any attribute in the search filter. A common search is with a ″changenumber >= *nnn*″ filter, where *nnn* is the largest changenumber value that was retrieved the previous time the search was done (the changenumber = *nnn* entry is retrieved again to ensure that the next part of the change log has not been trimmed).
- The change log entries matching the search filter are returned in increasing changenumber order.
- You cannot depend on there being change log entries for all consecutive change numbers. Some change numbers might be skipped.
- The change log (including the root entry) can be searched as long as change logging is enabled (the GDBM backend is configured), even if change logging is off.

Passwords in change log entries

To avoid including passwords in the **changes** attribute of a change log entry, the value of the **userpassword** and **racfpassword** attributes is replaced by `*ComeAndGetIt*`. You can use a search command to retrieve the password. For a RACF password, see Chapter 16, “Accessing RACF information,” on page 213 for more information.

Unloading and loading the change log

The unload utility (**tdbm2ldif**) cannot be used to unload the contents of the change log. You should use the search operation to do this. Change log entries cannot be loaded into the change log. Both the add operation and the bulkload utility (**ldif2tdbm**) fail when processing change log options.

Trimming the change log

When change logging is on, the LDAP server periodically trims the change log based on the limits set in the configuration file.

If a change log entry exceeds the age limit set using the **changeLogMaxAge** configuration option, it is removed from the log.

If the number of change log entries exceeds the limit set using the **changeLogMaxEntries** configuration option, the change log entries with the lowest changenumber values are removed until the number of entries is about 95% of the limit. For example, if **changeLogMaxEntries** is 1000 and the number of entries in the change log reaches 1001, the 51 lowest entries are deleted to reduce the number of entries to 950.

- The change log is checked for trimming when the server is started and when change log entries are added. The change log is also periodically trimmed, with a frequency determined by the server based on the change log limits and contents. The frequency cannot be directly modified.

- Trimming is only performed when change logging is on.

Change log information in the rootDSE entry

The following attributes in the rootDSE entry allow applications to determine the location of the change log and effectively use it. The attributes appear whenever change logging is enabled (the GDBM backend is configured), whether or not change logging is currently on.

changelog=CN=CHANGELOG
the location of the change log

firstchangenumber=*nnn*
the lowest change number currently in use in the change log. A zero indicates no change log entries.

lastchangenumber=*nnn*
the highest change number currently in use in the change log. A zero indicates no change log entries.

Multiserver considerations

Each server should have an identical GDBM backend configured. If a server does not have change logging on, changes made through that server are not logged, however, the change log can still be searched through that server. An LDAP Server with change logging on has to be running on the system where the change to be logged is made.

How to set up and use the LDAP server for logging changes

1. Update the LDAP server configuration file:
   a. Add the GDBM backend section, including a change log size and age limit if desired. The following example starts change logging using a change log with a maximum size of 1000 entries. Entries are automatically deleted when they become a day old.

      ```
      database gdbm GLDBGDBM
      dbuserid dbu1
      servername loc1
      changeLogging on
      changeLogMaxEntries 1000
      changeLogMaxAge 86400
      ```

   b. If you plan to log changes to RACF users, you must also:

      Add the SDBM backend section. Following is an example:

      ```
      database sdbm GLDBSDBM
      suffix cn=myRacf
      ```

      Enable the PC Callable support (used by RACF to add change log entries to the LDAP Server) by specifying the following option in the global section of the configuration file:

      ```
      listen ldap://:pc
      ```

   c. If you do not want to log changes to entries in a TDBM or GDBM backend, add the following option to the backend section:

      ```
      changeLoggingParticipant no
      ```

2. Create the DB2 database to be used by the change log. This involves updating and executing two SPUFI scripts. The database owner in the scripts must match the **dbuserid** value in the GDBM section of the configuration file. See “Creating the DB2 database and table spaces for TDBM or GDBM” on page 42 for more information.
3. If you plan to log changes to RACF users, perform the RACF configuration required to support creation of a password envelope, retrieval of the password envelope, and creation of an LDAP change log entry for changes to a RACF user. See *z/OS Security Server RACF Security Administrator's Guide* for more information.
4. Restart the LDAP server. You should receive a message similar to the following:

```
GLD0244I Change logging is enabled
Logging started status (0 = off, 1 = on): 1
Limit in seconds on age of change log entries (0 = no limit): 86400
Limit on the number of change log entries (0 = no limit): 1000
Current number of change log entries: 0
First change number in use: 0
Last change number in use: 0
```

If the LDAP server fails to configure the change log during startup, the following message is issued:

```
GLD0245I Change log configuration has failed and change logging is not enabled.
```

5. At this point, change logging is started. Depending on your configuration, a change to a RACF user or to a TDBM or GDBM entry will result in the creation of a change log entry in the LDAP server.
6. If desired, modify the ACL on the change log root entry, cn=changelog, for your usage of the change log. The initial ACL restricts client access to the change log to the LDAP administrator.

   For example, to give read access to the change log to RACF user CLREADER, create an ldif file, cl.ldif, similar to the following:

```
dn: cn=changelog
changetype: modify
add: aclentry
aclentry:access-id:racfid=clreader,profiletype=user,cn=myRacf:normal:rsc:
 sensitive:rsc:critical:rsc:system:rsc
-
```

   You should then modify the change log ACL by issuing a modify command similar to the following:

```
ldapmodify  -p nnn  -D adminDn  -w adminPw   -f cl.ldif
```

7. You can search, delete, and modify (to a limited extent) change log entries using the LDAP client interfaces and command line utilities. In particular, all change log entries can be viewed using a search similar to the following:

```
ldapsearch -p nnn  -D adminDn -w adminPw  -b "cn=changelog" "objectclass=*"
```

   Part of the output from this search would look like:

```
cn=changelog
objectclass=top
objectclass=container
cn=changelog

CHANGENUMBER=1,CN=CHANGELOG
objectclass=CHANGELOGENTRY
objectclass=IBM-CHANGELOG
objectclass=TOP
changenumber=1
targetdn=RACFID=U2,PROFILETYPE=USER,cn=myRacf
changetime=20030611204814.257756Z
changetype=MODIFY
changes=replace: racfPassword
racfPassword: *ComeAndGetIt*
-

ibm-changeinitiatorsname=RACFID=SUSET3,PROFILETYPE=USER,cn=myRacf
```

8. If the **changes** attribute of a change log entry contains either of the following lines:

```
racfPassword: *ComeAndGetIt*
userPassword: *ComeAndGetIt*
```

   then a password in the RACF user profile or TDBM entry was changed. See “Passwords in change log entries” on page 311 for information on retrieving passwords.

   Note: the password envelope is a binary value that is base-64 encoded

9. The LDAP root DSE entry contains useful information about the LDAP change log, including its suffix, and the lowest and highest change numbers currently in use. A command similar to the following one obtains this information:

```
ldapsearch -p nnn  -D adminDn -w adminPw  -V 3 -s base -b "" "objectclass=*"
```

Part of the output from this search would look like:

```
changelog=CN=CHANGELOG
firstchangenumber=1
lastchangenumber=202
```

Note: The LDAP server occasionally skips one or more change numbers, so it cannot be assumed that there is a change log entry for every number between 1 and 202. In addition, skips are created if you delete a change log entry that does not have the lowest number. Change numbers that are generated by the LDAP server are not guaranteed to be consecutive, but will always increase.

Chapter 26. Referrals

Referrals provide a way for servers to refer clients to additional directory servers. With referrals you can:
- Distribute namespace information among multiple servers
- Provide knowledge of where data resides within a set of interrelated servers
- Route client requests to the appropriate server

Following are some of the advantages of using referrals:
- Distribute processing overhead, providing primitive load balancing
- Distribute administration of data along organizational boundaries
- Provide potential for widespread interconnection, beyond an organization's own boundaries.

This chapter describes how to use the **referral** object class and the **ref** attribute to construct entries in an LDAP directory server containing references to other LDAP directory servers. Also described in this chapter is how to associate multiple servers using referrals and an example of associating a set of servers through referrals and replication (see Chapter 23, "Replication," on page 289).

In z/OS LDAP, referral entries are only supported in the TDBM (DB2-based) backend. The default referral can be used with any type of backend.

Using the referral object class and the ref attribute

The **referral** object class and the **ref** attribute are used to facilitate distributed name resolution or to search across multiple servers. The **ref** attribute appears in an entry named in the referencing server. The value of the **ref** attribute points to the corresponding entry maintained in the referenced server. While the distinguished name (DN) in a value of the **ref** attribute is typically that of an entry in a naming context below the naming context held by the referencing server, it is permitted to be the distinguished name of any entry. A multi-valued **ref** attribute may be used to indicate different locations for the same resource. If the **ref** attribute is multi-valued, all the DNs in the values of the **ref** attribute should have the same value.

Creating entries

Following is an example configuration that illustrates the use of the **ref** attribute.

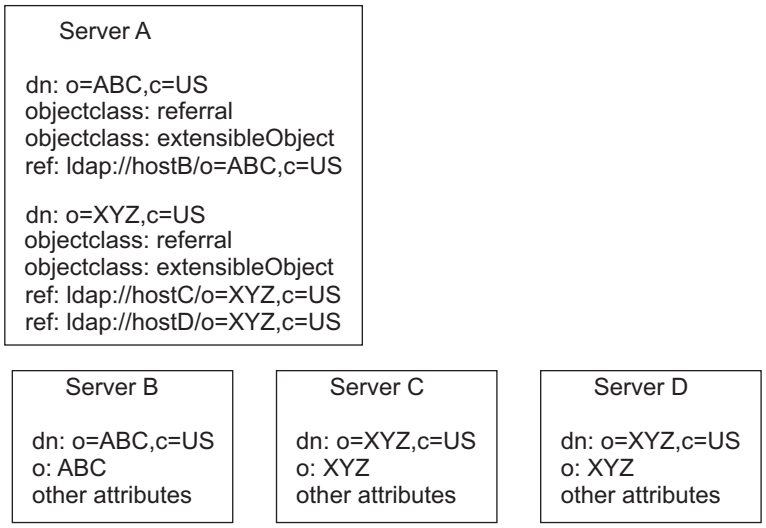


Figure 55. Example using ref attribute

In the example, Server A holds references to two entries: o=ABC,c=US and o=XYZ,c=US. For the o=ABC,c=US entry, Server A holds a reference to Server B and for the o=XYZ,c=US entry, Server A holds references to two equivalent servers, Server C and Server D.

The recommended setup of referrals is to structure the servers into a hierarchy based on the subtrees they manage. Then, provide "forward" referrals from servers that hold higher information and set the default referral to point back to its parent server.

Associating servers with referrals

In order to associate servers through referrals:
- Use referral objects to point to other servers for subordinate references
- Define the default referral to point somewhere else, typically to the parent server

These steps are defined below.

Pointing to other servers

Use referral objects to point to the other servers for subordinate references. That is, portions of the namespace below this server which it does not service directly.

Referral objects, like other objects, go in the TDBM backend. Referral objects consist of:

dn Specifies the distinguished name. It is the portion of the namespace served by the referenced server.

objectclass
Specifies **referral**. For entries in TDBM, also include the object class **extensibleObject**.

ref Specifies the LDAP URL of the server. This URL should consist of the **ldap://** or **ldaps://** identifier, the *hostname*:*port*, and a DN. The DN requires a slash (/) before it to delimit it from the *hostname*:*port*, and should match the DN of the referral object. The **ref** attribute may be multi-valued, with each value specifying the LDAP URL of a different server. When multiple values are used, each LDAP URL should contain the same DN, and each server should hold equivalent information for the portion of the namespace represented by the DN.

Following is an example:

```
dn:           o=IBM,c=US
objectclass:  referral
objectclass: extensibleObject
ref:          ldap://Host1:389/o=IBM,c=US
ref:          ldap://Host2:389/o=IBM,c=US
ref:          ldap://Host3:1389/o=IBM,c=US
```

The server can have any number of referral objects within its database. However, the objects must essentially be descendents of its suffix.

Defining the default referral

Define the default referral to point to another server which services other portions of the namespace unknown to the referencing server. The default referral can be used to point to:
- The immediate parent of this server (in a hierarchy)
- A "more knowledgeable" server, such as the uppermost server in the hierarchy
- A "more knowledgeable" server which possibly serves a disjoint portion of the namespace.

The default referral goes in the configuration file and not the backend. The default referral is described in the configuration file with the **referral** keyword and an LDAP URL. Multiple default referrals may be specified. However, each one specified is considered equivalent; that is, each server referenced by a default referral should present the same view of the namespace to its clients.

Note: The default referral LDAP URL does not include the DN portion. It should just have the **ldap://** identifier and the *hostname*:*port*. For example:

```
referral  ldap://host3.ibm.com:999
```

See “Operating in multi-server mode without dynamic workload management enabled” on page 84 and “Operating in multi-server mode with dynamic workload management enabled” on page 85 for information about setting up multiple default referrals.

SSL/TLS note: A non-secure client referral to a secure port is not supported. Also, a secure client referral to a non-secure port is not supported.

Processing referrals

When clients request information from servers which do not hold the needed data, servers can pass back referral URLs which indicate one or more other servers to contact. The clients can then request the information from the referenced server. The z/OS client API, by default, chases referrals returned from servers. However, client applications can suppress referral chasing through the **ldap_set_option()** API. This option’s scope is the LDAP handle, so a client could open multiple connections to one or more servers, some of which would chase referrals automatically, and some of which would not.

Servers present the referral URLs differently depending on the LDAP protocol version being used by the client. Referrals are presented to LDAP Version 2 clients in the error string, as the protocol does not provide a specific mechanism for indicating referrals. In LDAP Version 3, protocol elements are specifically defined to allow servers to present referral information to clients.

Using LDAP Version 2 referrals

LDAP Version 2 referrals are presented as part of the error string passed back to the client. Since clients do not generally examine the error string on results indicating **LDAP_SUCCESS**, some other error code is needed. Thus, on wholly successful operations, the server passes back a result of **LDAP_PARTIAL_RESULTS** instead of **LDAP_SUCCESS** to indicate the presence of referral information. On any result other than **LDAP_SUCCESS**, referral information might also be present in the error string. Note that some servers might return a result of **LDAP_PARTIAL_RESULTS** with no referral information if the server does not have a default referral defined and the client makes a request for a portion of the namespace outside the server (and not below it in the hierarchy).

The referral information in the error string looks like this:

```
Referral:\n
ldap://hostname:port/DN\n
...
```

where `Referral:` is followed by a new line character (`\n`) and `ldap://hostname:port/DN\n` is an LDAP URL followed by a new line character. The ellipses (`...`) indicate a list of multiple referrals; that is, more LDAP URLs followed by new line characters. Multiple referrals are only presented for partial search results when it is necessary to contact more than one additional server to complete the entire request. This would indicate that multiple referral objects were found in the referencing server that matched the search criteria. The client contacts every server presented in the list to continue the search request. For referral objects that have multi-valued **ref** attributes, the server sends only one of the LDAP URLs to a client using LDAP Version 2 protocol. This is because there is no provision for distinguishing between equivalent servers to contact (as indicated by multi-valued **ref** attributes) and multiple servers which must be contacted to complete a search request.

Limitations with LDAP Version 2 referrals

One limitation of LDAP Version 2 referrals is the lack of support for alternate referrals. As noted earlier, a referral object can have a multi-valued **ref** attribute which indicates different servers which hold equivalent information. As will be seen below, it is only possible for the LDAP Version 2 referral mechanism to contain

a single list of referrals for a given request. Thus, one cannot tell if it is a list of servers, all of which must be contacted to complete a request, a list of equivalent servers of which only one should be contacted, or a combination of both.

A second limitation of referrals in LDAP Version 2 is that operations can sometimes be ambiguous in their intent regarding whether the operation was targeted for "real" objects in the namespace, as opposed to the referral objects themselves. For searches, referral objects are only presented as referrals, since the usual intent of a search is to look at the real objects in the namespace. Server administrators must therefore use other means to examine existing referral objects, such as examining the database, or reviewing **tdbm2ldif** output. For update operations, default referrals for upward references are presented as referrals, so that read-only replica servers can forward update operations to the master replica. However, subordinate references indicated by a referral object are not followed for update operations, rather they operate on the referral object itself. This is necessary to allow an administrator the ability to delete or modify existing referral objects. Erroneous changes caused by misdirected update operations are generally avoided through access protection and schema rules.

Using LDAP Version 3 referrals

In LDAP Version 3, referrals are defined as part of the protocol. The LDAP Version 2 limitations mentioned above are overcome by elements of the protocol and extensions to the protocol. There are two methods of passing back referral information in the LDAP Version 3 protocol: referrals and search continuation references.

A result code of **LDAP_REFERRAL** is presented by the server to indicate that the contacted server does not hold the target entry of the request. The referral field is present in the result message and indicates another server (or set of servers) to contact. Referrals can be returned in response to any operation request except unbind and abandon which do not have responses. When multiple URLs are present in a given referral response, each one must be equally capable of being used to progress the operation.

A referral is not returned for a one-level or subtree search in which the search scope spans multiple naming contexts, and several different servers would need to be contacted to complete the operation. Instead, one or more search continuation references are returned. Search continuation references are intermixed with returned search entries. Each one contains a URL to another server (or set of servers) to contact, and represents an unexplored subtree of the namespace which potentially satisfies the search criteria. When multiple URLs are present in a given search continuation reference, each one must be equally capable of being used to progress the operation. By using a separate response for each unexplored subtree, LDAP Version 3 overcomes the limitation of LDAP Version 2, allowing alternate, equivalent servers to be included in each response.

As mentioned earlier, the other limitation in LDAP Version 2 referral processing is related to the inability of a client to specify whether a request was targeted for a normal object or a referral object. For LDAP Version 3, this difficulty is overcome with a protocol extension in the form of the **manageDsaIT** control. (Appendix D, "Supported server controls," on page 459 describes **manageDsaIT** in detail.) For typical client requests where the control is absent, whenever the server encounters an applicable referral object while processing the request, either a referral or search continuation reference is presented. When the client request includes this control, the server does not present any referrals or search continuation references, but instead treats the referral objects as normal objects. In this case, even superior references through the use of default referrals are suppressed. The shipped command line utilities support the **-M** option to indicate that the requestor is managing the namespace, and therefore wishes to examine and manipulate referral objects as if they were normal objects. An exception to the processing described above is that referral objects are always treated as normal entries during the second phase of a persistent search, even if the **manageDsaIT** control is not specified on the persistent search request. See "PersistentSearch" on page 461 for more information.

Example: associating servers through referrals and replication

Following are the steps involved in distributing the namespace using referrals.

1. Plan your namespace hierarchy.

```
country - US
company - IBM, Lotus
organizationalUnit - IBM Austin, IBM Endicott, IBM HQ
```

2. Set up multiple servers, each containing portions of the namespace.

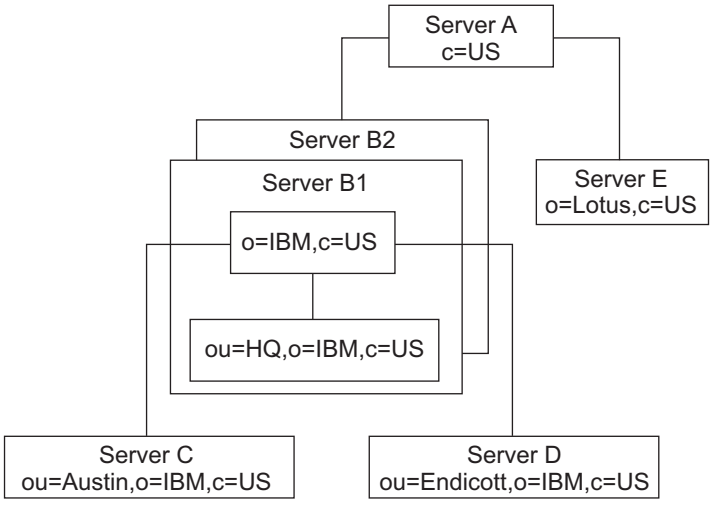


Figure 56. Setting up the servers

Following is a description of each server:

Server A
: Perhaps just a server used to locate other servers in the US. With no other knowledge, clients can come here first to locate information for anyone in the US.

Server B1
: A hub for all data pertaining to IBM in the US. Holds all HQ information directly. Holds all knowledge (referrals) of where other IBM data resides.

Server B2
: A replica of Server B1.

Server C
: Holds all IBM Austin information.

Server D
: Holds all IBM Endicott information.

Server E
: Holds all Lotus™ information.

3. Set up referral objects to point to the descendents in other servers.

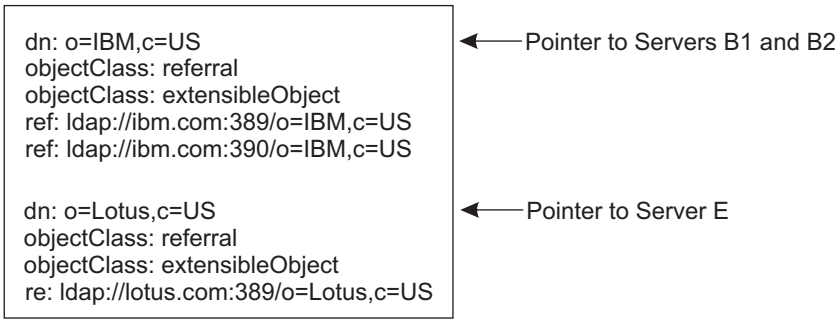


Figure 57. Server A database (LDIF input)

4. Servers can also define one or more default referrals which point to “more knowledgeable” servers for anything that is not underneath them in the namespace.

   The default referrals go in the configuration file, not the backend.

   Note: The default referral LDAP URLs do not include the DN portion.

```
#    General Section

referral              ldap://ibm.com:389
referral              ldap://ibm.com:390

listen                ldap://:789
.
.
.
# tdbm database definitions
database              tdbm GLDBTDBM
suffix                “ou=Endicott,o=IBM,c=US”
```

Figure 58. Server D configuration file

5. Putting it all together.

   Figure 59 on page 321, Figure 60 on page 322, and Figure 61 on page 323 show these same six servers, showing the referral objects in the database as well as the default referrals which are used for superior references. Also included in Servers B1 and B2 are sample definitions for replication, setting up Server B2 as a replica of Server B1. This ensures that these two servers will remain identical.

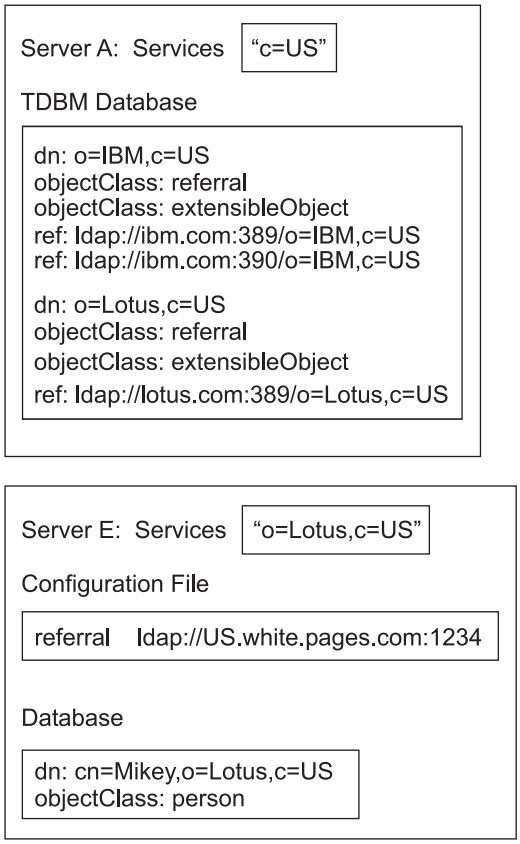


Figure 59. Referral example summary (servers A and E)

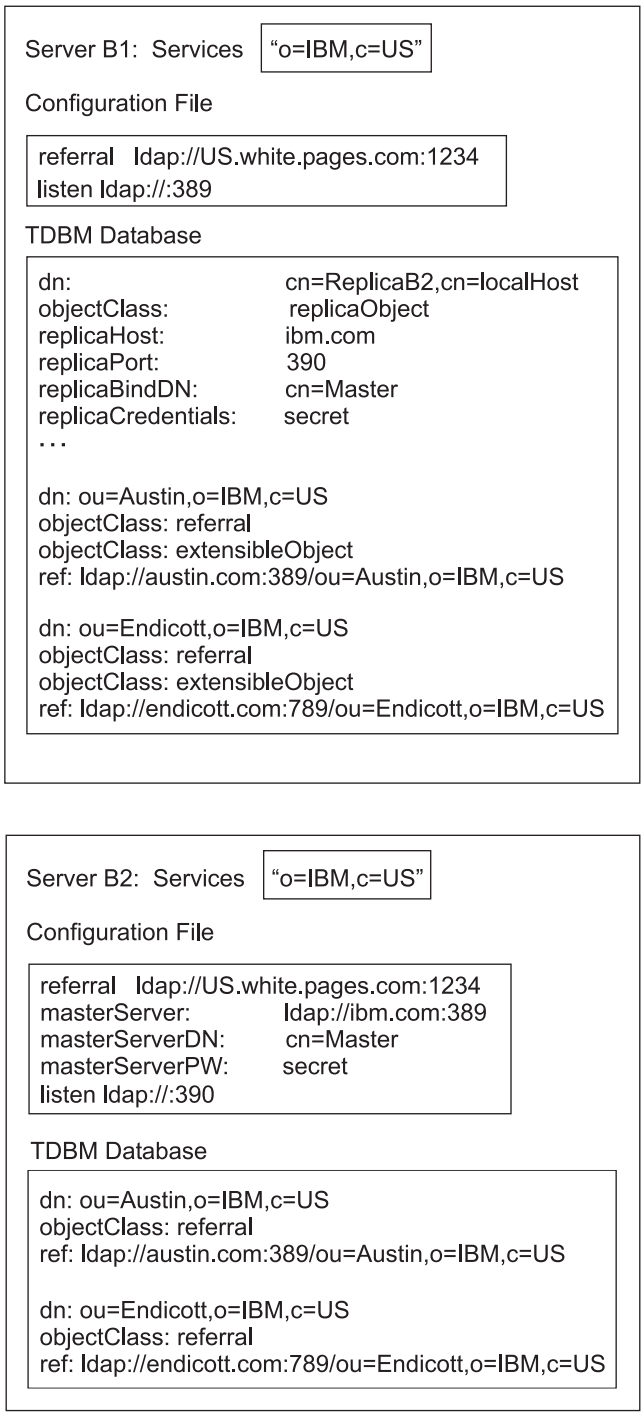


Figure 60. Referral example summary (servers B1 and B2)

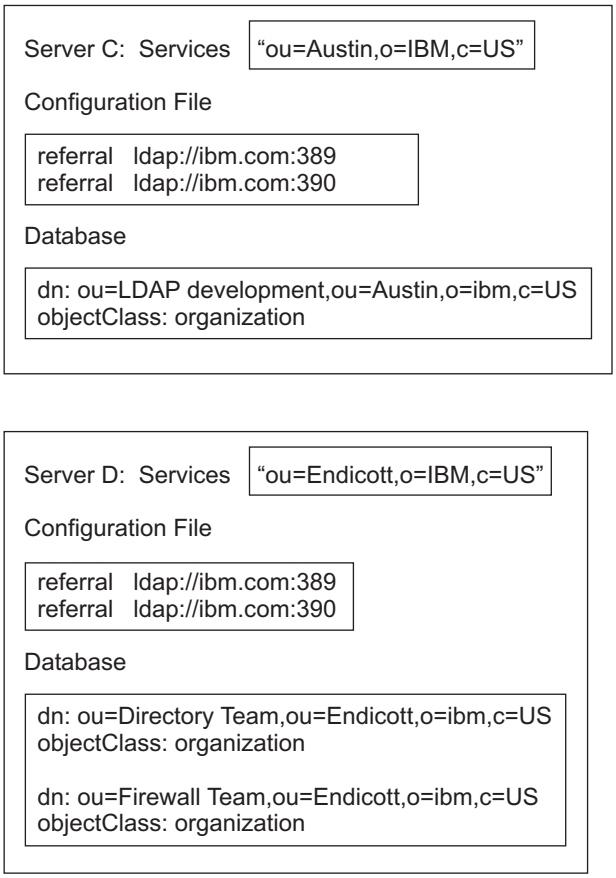


Figure 61. Referral example summary (servers C and D)

Chapter 27. Organizing the directory namespace

Directory services are meant to help organize the computing environment of the enterprise. To do this, directory services are meant to be used to help find all the resources at one's disposal. Information that is typically found in a directory consists of configuration information for services offered in the enterprise, locating information for people, places, and things in the enterprise, as well as descriptive information about services and resources available in the enterprise. The directory service should be thought of as the spot that can be queried to find whatever is desired in the enterprise.

When designing the format and organization of the directory service for an enterprise, the intended usage scenarios should be considered. These usage characteristics can have an impact on how the directory namespace should be organized so as to offer reasonable performance.

There are two general areas of directory namespace design to be considered. First, the types of information and the layout of where that information will be placed in the directory namespace must be determined. Additional information types can be added at a later date, but there should be some overall design of where in the directory namespace these types of information should be placed. Second, based on the usage characteristics of the users in the enterprise, the number of distinct directory servers and the namespace subtree or subtrees that they support must be considered.

As an example, consider an enterprise that consisted of two physical locations, one in Los Angeles, CA and one in New York City, NY. People in New York City access information about people, places, and things in Los Angeles often, while the people in Los Angeles rarely access information items in New York City. To offer good performance for both locations, a separate directory server could be installed and run in each location. These LDAP servers would manage information about the people, places, and things that reside in their respective locations. In addition, because the New York City personnel access information about things in the Los Angeles location, the information from the Los Angeles LDAP server could be replicated to an additional LDAP server at the New York City LDAP server. This would allow the New York City personnel to access information about the Los Angeles location by contacting a local server. In Los Angeles, however, directory requests about items in the New York City portion of the enterprise namespace are redirected (that is, referred) to the New York City LDAP server for the information. This would save managing a replicated set of information at the expense of slightly longer access times on the less-requested information.

The next two sections discuss information layout in the namespace and partitioning an enterprise namespace across multiple LDAP servers. These sections are followed by a small example.

Information layout

A directory is meant to provide information about people, places, and things in the enterprise. The most direct use of a directory is to hold information on how to contact other people in the enterprise. This has commonly been known as the *internal phone book*. With the widespread enhancements in technology, people are now more accessible than ever. We have pagers, answering machines, cellular phones, and e-mail. In trying to communicate with someone we might need to know about all of this information. Modeling a person object class based on the attributes about a person that are important to others in the enterprise is an easy way to support an online *internal phone book* using an LDAP directory service. In addition to people, different organizations within an enterprise can also be modeled by creating new object classes and attribute types. This would allow storage in the LDAP directory of locating information for useful services in the organization like benefits, travel reservations, and human resources.

Another application of directory services is the ability to model or store information about places. A place could be a conference room, which might have attributes of **numberOfSeats**, **projectorType**, **phoneNumber**, **calendarLocation**, **dataPortType**, **officeNumber**, and **buildingNumber**. Using this method, different conference rooms within a company could be located and compared. Another example of

a place would be the whole site in which employees work. An object class for a site LDAP directory entry might be made up of **streetAddress**, **generalManagerDN**, **siteMap**, and **cafeteriaLocation**.

Things abound within the enterprise. Under this category falls computers, copiers, FAX machines, printers, and computer software, as well as configuration information for servers that use an LDAP directory service. Each of these can be modeled with attribute types used inside object classes specific to the device or program.

In laying out where entries should appear in the directory hierarchy, by far the most common method of naming things is to start with the country in which the company is organized, followed by the name of the company, treated as an organization attribute type. Thus, the top level suffix for LDAP directory service names for entries within the company sometimes follows the form: `o=CompanyName, c=US` (for US-based companies). Alternately, the top level suffix may follow the domain form, for example: `dc=CompanyName,dc=com`. Below this suffix it is common for organizational unit object classes to be used to represent departments or sites within an organization. Below these organizational entries the actual entry representing a person, place, or thing would be defined. When organizing the information layout for the namespace, the intended usage should be considered to ensure the best performance.

Example of building an enterprise directory namespace

Let us look at an example configuration that exhibits the features available with the z/OS LDAP server. To set the stage, we will consider a moderately sized company that has personnel working in three locations across the United States. Big Company, Inc. has corporate headquarters in Chicago, IL, and two satellite facilities, one in Los Angeles, CA and the other in New York City, NY. The information technology staff would like to make available information about all of the company's computing and office services using an LDAP directory. In order to facilitate local modifications as necessary of the information in the directory, as well as provide improved response time for accessing local information, each site will have an LDAP server running. The server running at each site will be responsible for managing the directory information that pertains to that site.

The first thing to do is determine the name of the root of the directory namespace for Big Company, Inc. Typically, the name for the company will consist of the country of origin along with the company's given name. In LDAP directory terminology, the company is an organization. In this example, we chose:

```
o=Big Company, c=US
```

as the company's name is Big Company and is located in the United States. Choosing a name of this format helps ensure that when a global namespace coordinator is established, the company's chosen *root* will fit nicely into the overall directory namespace.

Next to choose are the names of the three locations under which the directory information is stored. At this point, the namespace could be organized in a number of ways. One way would be to organize by functional unit (regardless of location). This model is useful if individuals (or computers, or other equipment or services) typically remain with the functional unit as opposed to being tied to the individual or physical location. Another way would be to organize based on the physical locations of the parts of the organization. This is useful if the people, places, and things to be stored in the directory typically do not move between locations. This latter approach will be used in the example. So, with three locations, three names are defined below the overall company distinguished name:

```
ou=Los Angeles, o=Big Company, c=US
ou=Chicago, o=Big Company, c=US
ou=New York City, o=Big Company, c=US
```

Since separate LDAP servers will be established at each location, these names represent the root of the subtree stored and managed by the directory server at each location.

For administration, each site will have a different directory administrator. To define this administrator, an administrator distinguished name and password need to be defined for each location. To start, the following names will be used:

```
AdminDN "cn=Administrator, ou=Los Angeles, o=Big Company, c=US"

AdminDN "cn=Administrator, ou=Chicago, o=Big Company, c=US"

AdminDN "cn=Administrator, ou=New York City, o=Big Company, c=US"
```

Since the Chicago location is also the corporate headquarters, the LDAP directory at this location will be used to store information about the entire company as well as information about the Chicago site.

We now have enough information to set up the base configuration files for each of the three LDAP servers that will be used to supply this information. Following are the files needed to set up the LDAP servers on each site. Note that what is shown is the minimal setup required. Other options could be specified in addition to these. See “Creating the slapd.conf file” on page 53 for configuration file options.

```
# Configuration file for the Chicago LDAP server
adminDN "cn=Administrator, ou=Chicago, o=Big Company, c=US"

database tdbm GLDBTDBM
suffix "o=Big Company, c=US"

dbuserid user1
databasename dbldap1
servername loc1
# end of configuration file
```

Figure 62. Chicago base configuration

```
# Configuration file for the Los Angeles LDAP server
referral ldap://ldap.chicago.bigcompany.com
adminDN "cn=Administrator, ou=Los Angeles, o=Big Company, c=US"

database tdbm GLDBTDBM
suffix "ou=Los Angeles, o=Big Company, c=US"

dbuserid user2
databasename dbldap2
servername loc1
# end of configuration file
```

Figure 63. Los Angeles base configuration

```
# Configuration file for the New York City LDAP server
referral ldap://ldap.chicago.bigcompany.com

adminDN "cn=Administrator, ou=New York City, o=Big Company, c=US"

database tdbm GLDBTDBM
suffix "ou=New York City, o=Big Company, c=US"

dbuserid user3
databasename dbldap3
servername loc1
# end of configuration file
```

Figure 64. New York City base configuration

The referral line indicates the default place to refer connecting clients when the LDAP server does not contain the information requested by the client. It is called the *default referral*. It is in the form of an LDAP URL. After the scheme name (`ldap`), the LDAP URL contains a TCP/IP DNS host name for another LDAP server. In this example, it is assumed that the TCP/IP host on which the Chicago LDAP server is running is `ldap.chicago.bigcompany.com`. The Chicago LDAP server does not have a default referral defined. This keeps directory searches from inadvertently going over the Internet from within the company.

The `adminDN` line indicates the distinguished name that should be used to connect to the LDAP server in order to have complete control over the data content held by the LDAP server.

The `database` line indicates that all following lines pertain to the TDBM storage method. The suffix line indicates what part of the namespace is contained in this server.

The next lines are used to connect to DB2 as well as identify the DB2 tables (named based on userid) that were defined for the LDAP server (see “Creating the DB2 database and table spaces for TDBM or GDBM” on page 42).

After these files have been created and the SPUFI scripts have run successfully, one or more of the LDAP servers can be started. However, there will be no initial data in the DB2 tables. The next section introduces a load utility that can be used to load entries into the LDAP server.

Priming the directory servers with information

There are several methods of adding entries to an LDAP directory server: using the TDBM load facility (**ldif2tdbm**), using the **ldapadd** and **ldapmodify** tools, or using the LDAP C language API and the LDAP protocol. If you are not using **ldif2tdbm**, it is recommended that at least the top levels of directory information be loaded first into the database. This provides a base from which to add more entries into the directory namespace. If you are using **ldif2tdbm**, add the top levels and the additional entries at the same time to achieve the best load performance.

The **ldif2tdbm** utility program for TDBM can do the check and prepare phases while the LDAP server is running, but the actual loading of data cannot be done while the LDAP server is running. The load utility program makes direct updates to the directory database tables without using the LDAP protocol. This method results in faster loads of large amounts of directory information. The load utility takes as input a sequential file that contains the data describing directory entries to be added into the directory namespace. The format of the information in the sequential (HFS) file is LDAP Interchange Format (LDIF).

Using LDIF format to represent LDAP entries

The LDAP Data Interchange Format (LDIF) is used to represent LDAP entries in a simple text format. An LDIF file contains groups of attribute information which will be treated as an entry to be added to the directory. The general format of an LDIF entry is:

```
dn: distinguished name
attrtype1: attrvalue1
attrtype2: attrvalue2
...
```

Each line in the LDIF file must begin in column 1. However, to continue a line, start the next line with a single space or tab character. For example:

```
dn: ou=departments, ou=New York City, o=Big Co
 mpany, c=US
```

Multiple attribute values are specified on separate lines. For example:

```
objectclass: organizationalunit
ou: departments
```

Note about editing LDIF files

Be aware that some editors, including **oedit**, place blank spaces at ends of all empty lines within a file. A blank space at the beginning of a line signifies continuation of the entry. The blank lines used to separate entries may be treated as continuations of an attribute value instead of separators if an editor has modified the LDIF file. Also, be aware that some editors, including oedit, delete blanks at the end of a line that is not empty. This can change the value of an attribute, especially if that value is continued on the next line.

If an *attrvalue* contains a nonprinting character, or begins with a space or a colon (:), the *attrtype* is followed by a double colon (::) and the value is encoded in base64 notation. For example, the value:

```
" begins with a space"
```

would be encoded like this:

```
cn:: IGJlZ2lucyB3aXRoIGEgc3BhY2U=
```

Multiple entries within the same LDIF file are separated by blank lines. Here is an example of an LDIF file containing three entries.

```
dn: ou=New York City, o=Big Company, c=US
objectclass: organizationalunit
ou: New York City

dn: ou=fax machines, ou=New York City, o=Big Company, c=US
objectclass: organizationalunit
ou: fax machines

dn: ou=computers, ou=New York City, o=Big Company, c=US
objectclass: organizationalunit
ou: computers
```

Note: Trailing spaces are not trimmed from values in an LDIF file. Also, multiple internal spaces are not compressed. If you do not want them in your data, do not put them there.

Multiple attribute values for the same attribute type are specified on multiple lines within the specification of a directory entry. For example:

```
dn: cn=John Doe, ou=New York City, o=Big Company, c=US
objectclass: person
cn: John Doe
phonenumber: 555-1111
phonenumber: 555-2222
sn: Doe
```

Generating the file

A file is typically generated using an existing source of information and some tools to format the data into the LDIF format. Note that the order of entries in the LDIF file is important. In order for an entry specified in the LDIF file to be successfully added to the directory, its parent entry must first exist in the directory namespace. For this reason, the top level entries in the directory namespace subtree that the particular LDAP server will support must be first in the LDIF file.

For our example, we will define just the a minimal set of entries to get the directory server useful at each location. This will include two referral entries for the Chicago location. The meaning of these entries will be discussed in more detail in the following sections.

Here is the base set of LDIF files to set up the directory namespace at each location. For the Los Angeles location:

```
dn: ou=Los Angeles, o=Big Company, c=US
objectclass: organizationalunit
ou: Los Angeles

dn: cn=Administrator, ou=LosAngeles, o=Big Company, c=US
objectclass: person
cn: Administrator
sn: Administrator
userpassword: xxxxx

dn: ou=fax machines, ou=Los Angeles, o=Big Company, c=US
objectclass: organizationalunit
ou: fax machines

dn: ou=computers, ou=Los Angeles, o=Big Company, c=US
objectclass: organizationalunit
ou: computers

dn: ou=departments, ou=Los Angeles, o=Big Company, c=US
objectclass: organizationalunit
ou: departments
```

For the New York City location:

```
dn: ou=New York City, o=Big Company, c=US
objectclass: organizationalunit
ou: New York City

dn: cn=Administrator, ou=New York City, o=Big Company, c=US
objectclass: person
cn: Administrator
sn: Administrator
userpassword: xxxxx

dn: ou=fax machines, ou=New York City, o=Big Company, c=US
objectclass: organizationalunit
ou: fax machines

dn: ou=computers, ou=New York City, o=Big Company, c=US
objectclass: organizationalunit
ou: computers

dn: ou=departments, ou=New York City, o=Big Company, c=US
objectclass: organizationalunit
ou: departments
```

For the Chicago location:

```
dn: o=Big Company, c=US
objectclass: organization
o: Big Company

dn: ou=Los Angeles, o=Big Company, c=US
objectclass: referral
objectclass: extensibleObject
ref: ldap://ldap.losangeles.bigcompany.com/ou=Los Angeles,o=Big Company,c=US

dn: ou=New York City, o=Big Company, c=US
objectclass: referral
objectclass: extensibleObject
ref: ldap://ldap.newyorkcity.bigcompany.com/ou=New York City,o=Big Company,c=US

dn: ou=Chicago, o=Big Company, c=US
objectclass: organizationalunit
ou: Chicago

dn: cn=Administrator, ou=Chicago, o=Big Company, c=US
objectclass: person
```

```
cn: Administrator
sn: Administrator
userpassword: xxxxx

dn: ou=fax machines, ou=Chicago, o=Big Company, c=US
objectclass: organizationalunit
ou: fax machines

dn: ou=computers, ou=Chicago, o=Big Company, c=US
objectclass: organizationalunit
ou: computers

dn: ou=departments, ou=Chicago, o=Big Company, c=US
objectclass: organizationalunit
ou: departments
```

These files will now be used with a load facility. For a TDBM backend database, use the following command from the z/OS shell:

```
ldif2tdbm -o output.datasetHLQ -i ldif.filename -f config.filename -cpl
```

You can also run **ldf2tdbm** from TSO or from the batch environment.

After running this command on each respective directory server system, the directory namespace will be formed and the servers can now be used to hold and supply information.

Two entries added to the Chicago location directory server database deserve some special attention. These are the referral objects that were in the LDIF file for the Chicago location. Notice that the referral objects have the identical distinguished name as the root of the LDAP directory namespace that is served by the Los Angeles and New York City servers. These entries, coupled with the default referral specification in the configuration file for the LDAP servers in Los Angeles and New York City, enable searches of the Big Company namespace to originate at any of the three directory servers and resolve to the correct server to obtain the information.

A referral redirects a client request to a different LDAP server that can presumably handle the request (or refer the client to another server that can handle the request). In our example, if a client connects to the New York City server requesting a name that is under the Los Angeles portion of the namespace, the New York City server will send back a referral to the client based on the default referral. This will point the client at the Chicago directory server. The Chicago server will resolve the request down to the referral object for distinguished name `ou=Los Angeles, o=Big Company, c=US` and refer the client to the Los Angeles server. Finally, the client will contact the Los Angeles server and obtain the information requested.

Setting up for replication

As people start using the directory service in their daily routines at Big Company, Inc., the information technology staff notices that the people in New York City are doing a lot of work with the people in Los Angeles. So much, in fact, that an analysis of the TCP/IP traffic between New York City and Los Angeles shows that much of the traffic is directory access requests, presumably to look up phone numbers or FAX numbers for people in Los Angeles. The information technology staff decides to improve directory lookup response time, as well as lessen the directory lookup traffic between New York City and Los Angeles, by creating a replica of the Los Angeles directory server's information in New York City. This will allow local access to this information by the New York City users and cut down on the amount of requests from New York that must travel to Los Angeles to be completed.

Defining another LDAP server

To set up a replica of the LDAP server information in Los Angeles, a second LDAP server must be defined and started in New York City. This server can reside on the same system as the first LDAP server, though if this is chosen, the TCP/IP port that this replica server listens on must be different from the other LDAP server running on the system. As an alternative, the replica server could run on a different system,

allowing it to listen on the default LDAP port. The configuration file for the replica server in New York City will be very similar to the configuration files for the New York City server and the Los Angeles server. This configuration file must contain some additional items that pertain to replication. Here is what the contents of the New York City Los Angeles replica server should contain:

```
# Configuration file for the New York City Los Angeles replica LDAP server
referral ldap://ldap.chicago.bigcompany.com

listen ldap://:2001
adminDN "cn=Administrator, ou=Los Angeles, o=Big Company, c=US"

database tdbm GLDBTDBM
suffix "ou=Los Angeles, o=Big Company, c=US"

dbuserid user2
databasename dbldap
servername loc1

masterServer ldap://ldap.losangeles.bigcompany.com
masterServerDN "cn=Replicator, ou=Los Angeles, o=Big Company, c=US"
# end of configuration file
```

The additional lines at the end of the configuration file specify the only “user” that can update entries in the replica LDAP server. The values here must match the values entered at the “source” location when the replica is defined.

Preparing the replica

The next step is to get the LDAP replica primed with the existing information in the Los Angeles server and set up the Los Angeles server to replicate to the New York City replica. The set of steps to perform (described in “Populating a replica” on page 293) ensures that the replicas are in sync and that no update is lost during this synchronization. Once the replica is defined at the source location, updates to the directory information will be logged to be sent to the replica server when possible.

To initially synchronize the data between the LDAP master server and the LDAP replica server, perform the steps in “Populating a replica” on page 293.

While there are a number of manual steps to perform, there is a small consolation that the steps at different locations are not interleaved. All work can be done at the source location and then all work can be performed at the target (replica) location.

Resynching the replica and master servers

If it is noticed that a replica’s contents are out of sync with the information at the master server, the information can be resynched by following the steps shown in “Recovering from out-of-sync conditions” on page 299.

Notifying users of the replica

At this point, the New York City users can be notified that a second LDAP server is now available for their use. The notification should contain either the LDAP URL of the new LDAP replica server or the host name and port number of the LDAP replica server, as well as the base of the LDAP subtree that is published by the replica. As updates are made to the Los Angeles LDAP server, these updates will be propagated to the replica server in New York City. See Chapter 23, “Replication,” on page 289 for more details on replication.

What Big Company, Inc. now has in place is an Enterprise Directory service that can be used by whatever enterprise distributed processing tasks require lookup or configuration information. These enterprise distributed processing tasks and applications may require some changes to make use of the directory

service, but the result will be the ability to view, find, and modify the configuration of the enterprise by looking at and modifying the contents of the LDAP directory.

Chapter 28. Client considerations

When an LDAP application is communicating with an z/OS LDAP server, you should consider the following special topics:

- Root DSE
- Monitor Support
- CRAM-MD5 authentication support
- UTF-8 data over the LDAP Version 2 protocol
- Attribute types stored and retrieved in lowercase
- Abandon behavior
- Changed return codes
- Changed reason codes
- Reason codes

Root DSE

Following is an example of the information that the z/OS LDAP server will report on a search of the root DSE. A subset of these values may appear in your rootDSE based on the server configuration choices you have made.

```
supportedcontrol=2.16.840.1.113730.3.4.2
supportedcontrol=1.3.18.0.2.10.2
supportedcontrol=1.3.18.0.2.10.10
supportedcontrol=1.3.18.0.2.10.11
supportedcontrol=1.3.18.0.2.10.20
supportedcontrol=1.3.18.0.2.10.6
supportedextension=1.3.6.1.4.1.1466.20037
supportedextension=1.3.18.0.2.12.8
supportedextension=1.3.18.0.2.12.7
namingcontexts=cn=myRACF
namingcontexts=CN=CHANGELOG
namingcontexts=o=IBM,c=US
namingcontexts=secAuthority=Default
ibm-supportedcapabilities=1.3.18.0.2.32.19
ibm-supportedcapabilities=1.3.18.0.2.32.3
ibm-supportedcapabilities=1.3.18.0.2.32.31
ibm-supportedcapabilities=1.3.18.0.2.32.7
ibm-supportedcapabilities=1.3.18.0.2.32.33
ibm-supportedcapabilities=1.3.18.0.2.32.34
ibm-supportedcapabilities=1.3.18.0.2.32.30
ibm-supportedcapabilities=1.3.18.0.2.32.28
ibm-supportedcapabilities=1.3.18.0.2.32.24
ibm-enabledcapabilities=1.3.18.0.2.32.3
ibm-enabledcapabilities=1.3.18.0.2.32.7
ibm-enabledcapabilities=1.3.18.0.2.32.33
ibm-enabledcapabilities=1.3.18.0.2.32.34
ibm-enabledcapabilities=1.3.18.0.2.32.31
ibm-enabledcapabilities=1.3.18.0.2.32.28
ibm-enabledcapabilities=1.3.18.0.2.32.24
subschemasubentry=CN=SCHEMA,o=IBM,c=US
supportedsaslmechanisms=EXTERNAL
supportedsaslmechanisms=CRAM-MD5
supportedsaslmechanisms=DIGEST-MD5
supportedldapversion=2
supportedldapversion=3
ibmdirectoryversion=z/OS V1R6
ibm-sasldigestrealmname=MYHOST.IBM.COM
altserver=ldap://host2.ibm.com:999
ref=ldap://hostk.ibm.com:391
changelog=CN=CHANGELOG
firstchangenumber=24213
lastchangenumber=24322
```

Following are Object Identifiers (OIDs) for supported and enabled capabilities:

Short name	Description	OID assigned
Entry UUID	Identifies that this server supports the ibm-entryuuid operational attribute.	1.3.18.0.2.32.3
Max Age ChangeLog Entries	Specifies that the server is capable of retaining changelog entries based on age.	1.3.18.0.2.32.19
Monitor Operation Counts	The server provides new monitor operation counts for initiated and completed operation types.	1.3.18.0.2.32.24
TLS Capabilities	Specifies that the server is capable of doing TLS.	1.3.18.0.2.32.28
Kerberos Capability	Specifies that the server is capable of using Kerberos.	1.3.18.0.2.32.30
ibm-allMembers and ibm-allGroups operational attributes	Indicates that a backend supports searching on the ibm-allGroups and ibm-allMembers operational attributes.	1.3.18.0.2.32.31
Modify DN (subtree move)	Indicates that a subtree can be moved to another subtree, within a backend. This move uses a new superior. It can also use a new RDN.	1.3.18.0.2.32.33
Modify DN (subtree rename)	Indicates that a subtree can be renamed. The DN of each entry under the subtree will also be changed. This rename uses a new RDN but not a new superior.	1.3.18.0.2.32.34

Monitor Support

You can retrieve statistics from the server by issuing a search request with a search base of **cn=monitor** and a filter of (**objectclass=***). For details, see “Monitoring performance with **cn=monitor** search” on page 358.

CRAM-MD5 Authentication Support

CRAM-MD5 authentication is supported on the Tivoli® IBM Directory Server and client utilities. However, the way that it has been implemented on Tivoli IBM Directory Server is different than the z/OS LDAP server. Thus, this has resulted in differences between the Tivoli IBM Directory Server and the z/OS LDAP server client utilities. In order to perform a CRAM-MD5 authentication bind with the Tivoli IBM Directory Server client utilities to the z/OS LDAP server, you must specify the bind DN (authorization DN) with the **-D** option. The Tivoli IBM Directory Server client utilities do not support specifying an authentication identity (username) on a CRAM-MD5 bind. See Chapter 19, “CRAM-MD5 and DIGEST-MD5 Authentication,” on page 251 for more information on CRAM-MD5 and DIGEST-MD5 bind authentication on the z/OS LDAP server.

UTF-8 data over the LDAP Version 2 protocol

The LDAP Version 3 Protocol allows UTF-8 attribute values outside of the IA5 character set to be stored in the directory. This information must be able to be returned in some format to LDAP Version 2 clients. An additional server configuration file option, **sendV3stringsoverV2as**, which has the possible values **ISO8859-1** or **UTF-8**, can be used to indicate which format to use when sending this information over the Version 2 protocol.

Note: Different clients treat non-IA5 data differently over the Version 2 protocol. Refer to the documentation for the client APIs you are using for more information.

Attribute types stored and returned in lowercase

The LDAP server stores and returns attribute types in lowercase (normalized). For example, the attribute type "`productName`" is returned as "`productname`".

Abandon behavior

In releases of the LDAP server prior to z/OS V1R4, an abandon operation did not affect the processing of the targeted operation because all operations were handled by the server in a sequential order. By the time the abandon was received, the operation that was the target of the abandon had already completed.

In z/OS V1R4 and later releases, the LDAP server reads additional operations as they arrive as long as the connection is not a secure connection and the previous operation is not bind, unbind, or extended operation. This allows the LDAP server to process abandon operations as they are received and affect previously submitted operations.

Changed return codes

For information on changed LDAP return codes, see *z/OS Migration*.

Changed reason codes

The following reason code numbers were incorrect in the previous versions of this book.

Note: The actual reason code number and text has not changed.

Incorrect numbers	Correct numbers
R003116 - R003127	R003115 - R003126

For more information on changed LDAP reason codes, see *z/OS Migration*.

Reason codes

The **LDAPResult** construct is used by the LDAP protocol to return success or failure indications from servers to clients. This construct contains an error message field. Servers can optionally provide "human-readable" diagnostic information in this field. Depending on the location in the z/OS LDAP server where errors are detected, error messages generated may have the following format:

`R<numeric digits> <diagnostic information> <traceback information>`

where:

numeric digits
: Represents a specific reason code.

diagnostic information
: Provides details about the reason for the failure.

traceback information
: Is of the form (*file_identification* | *file_version* |*line_number*) and will assist you in diagnosing application or configuration problems.

Note the following regarding this error information:

- It is intended to be "human-readable" to assist in identifying problems detected by the server.
- It is not translated (English text only).
- It is not intended to be used as an application programming interface (API).

- Data returned may be changed by service or new releases of the product. (Again, it is not intended to be an API.)

The new LDAP reason codes for z/OS Version 1 Release 6 are: R000012 - R000015, R001073, R002019 - R002026, R003127, R004153 - R004174, R006016 - R006022, R007040 - R007052.

Following is the current list of reason codes and associated diagnostic information returned by the z/OS LDAP server.

R000001	**An attempt to allocate heap storage failed.**
R000002	**An attempt to allocate heap storage failed. Length:** *length.*
R000004	**An internal server error has been encountered.**
R000005	**Attempt to translate data from** *source_codepage* **to** *target_codepage* **failed with LDAP rc=0x***rc.*
R000006	**Error on BER encoding of** *BER_element***.**
R000007	**Error on BER encoding of attribute** *attr_type***.**
R000008	**Error on BER encoding of attribute** *attr_type* **and its values.**
R000009	**Encoding error on BER encoding value(s) of attribute** *attr_type***.**
R000010	**The filter used to search for the schema is not valid. Filter should be objectclass=subschema.**
R000011	**Error on BER decoding of attribute(s):** *attribute-name*
R000012	**Error writing BER of length:** *length*
R000013	*string*
R000014	*length string*
R000015	**The value specified was not valid.**
R000100	**The password has expired.**
R000101	**The new password is not valid.**
R000102	**The user ID has been revoked.**
R000103	**The user ID is not defined.**
R000104	**The password is not correct or the userid is not completely defined (missing password or uid).**
R000105	**A bind argument is not valid.**
R000106	**An unknown error occurred.**
R000107	**SDBMBeCapability: Constructor missing a required initializer.**
R000108	**Constructor call to entry_attr_merge_becap failed.**
R000109	**SDBMBeCapability::operator= not implemented.**
R000110	**SDBMBeCapability::operator== not implemented.**
R000111	**SDBMBeCapability::default constructor not implemented.**
R000112	**SDBMBeCapability::getNextVal: Bad cursor argument.**
R000113	**SDBMBeCapability::getNextVal: NULL cursor argument.**
R000114	**The realm portion of the value of attribute** *attr_type* **is not the RACF default realm.**
R000115	**There is no RACF default realm.**
R000116	**Cannot specify a value when deleting attribute** *attr_type***.**
R000117	**Cannot delete attribute** *attr_type***.**

R000118	**Cannot replace attribute** *attr_type*.
R000119	**Cannot add or replace attribute** *attr_type*.
R000120	**Cannot specify more than one value for attribute** *attr_type*.
R000121	**Cannot specify a value for attribute** *attr_type* **that is different than the value in the DN.**
R000122	**The value for attribute** *attr_type* **must be the DN of a user.**
R000123	**The value for attribute** *attr_type* **must be the DN of a group.**
R000124	**The value for attribute** *attr_type* **must be the DN of a user or a group.**
R000125	**Attribute** *attr_type* **is not supported.**
R000126	**Filter** *filter* **is not supported for this base.**
R000127	**Filter** *filter* **contains a type without a value.**
R000128	**Filter** *filter* **is not supported.**
R000129	**Value** *value* **is not supported for filter** *filter*.
R000130	**The syntax of filter** *filter* **is not valid.**
R000131	*DN* **is not a valid RACF DN.**
R000132	**Value for attribute** *attr_type* **cannot be more than** *length* **characters.**
R000133	**Value for attribute** *attr_type* **must be an integer less than** *size*.
R000134	**The RACF** *command* **command created to satisfy this request is too long, probably due to specifying a long filter or attribute value or too many attribute values.**
R000135	**Cannot perform this request on a reserved RACF DN,** *DN*.
R000136	**Operation not supported. The schema entry cannot be used in the compare operation.**
R000137	*DN* **is not a valid RACF DN for bind. Check that the syntax is correct and that it is a DN for a RACF user.**
R001001	*value* **is not a valid Generalized Time format value.**
R001002	**Initialization of required schema failed.**
R001003	**RDN specified for schema entry not valid.**
R001004	**Schema entry contained unknown information.**
R001005	**Duplicate value encountered:** *value.*
R001006	**Value not found in schema. Value:** *value.*
R001007	**Program used improperly set pointer.**
R001008	**Specified value did not match syntax.**
R001009	**Specified value did not match syntax. Value:** *value.*
R001010	**Specified value did not match syntax. Value:** *value*.
R001011	**Collective keyword not supported for attribute type:** *attr_type*.
R001012	**Attribute information not found for:** *attr_type*.
R001013	**Usage designated for superior attribute type not consistent with attribute:** *attr_type*.

R001014	**Type designated for superior object class:** *object_class* **not consistent with object class:** *object_class*.
R001015	**Cycle detected in** *SUP_hierarchy*.
R001016	*required_information* **missing for schema value:** *value*.
R001017	**Syntax/matching rule inconsistency for attribute type:** *attr_type*.
R001018	**Value specified is obsolete. Value:** *value*.
R001019	**Internal error during schema initialization.**
R001020	**User schema did not contain required value:** *value*.
R001021	**User schema did not contain required value:** *value* **for** *value*.
R001022	**User and required schema keyword values did not match for** *attr_type*.
R001023	**Building minimum schema failed.**
R001024	**Abstract object class:** *object_class* **can not be leaf.**
R001025	**Multiple structural object classes found as leaf:** *object_class*, *object_class*.
R001026	**Structural object class not found for entry.**
R001027	**Entry attempted to change base structural object class from:** *object_class* **to** *object_class*.
R001028	**More than one value specified for single valued attribute type:** *attr_type*.
R001029	**Must attribute type not found in entry:** *attr_type*.
R001030	**Entry contained attribute type not allowed by schema:** *attr_type*.
R001031	**Missing left parenthesis in schema value:** *value*.
R001032	**Missing right parenthesis in schema value:** *value*.
R001033	**Schema processing encountered an internal error.**
R001034	**An unsupported option was found for keyword:** *keyword* **in value:** *value*.
R001035	**End of data found when expecting more values in value:** *value*.
R001036	**Length value was not specified in value:** *value.*
R001037	**Non-numeric length specified in value:** *value.*
R001038	**Format of numeric object identifier not valid in value:** *value*.
R001039	**Unsupported non-alphabetic character** '*character*' **at offset** *offset* **in value:** *value*.
R001040	**Unsupported character** '*character*' **at offset** *offset* **in value:** *value*.
R001041	**Values not found where expected in value:** *value*.
R001042	**Missing a leading delimiter (***delimiter***) in a portion of the value:** *value*.
R001043	**Missing a trailing delimiter (***delimiter***) in a portion of the value:** *value*.
R001044	**A required blank is missing at offset** *offset* **in value:** *value*.
R001045	**Keyword:** *keyword* **not supported for value:** *value*.

R001046	**Missing closing quote for value:** *value*.
R001047	**Missing leading quote for value:** *value*.
R001048	**Missing closing brace for value:** *value*.
R001052	**Non-numeric characters found in an Integer value:** *value*.
R001053	**Integer value of length** *length* **exceeds the maximum allowable size (***size***).**
R001054	**DirEntry parameter initialized on input.**
R001055	**Specified attribute:** *attr_value* **not valid for schema entry.**
R001056	**Specified object class:** *object_class* **not valid for schema entry.**
R001057	**Schema entry missing required object class.**
R001058	**Schema entry missing required** *value*.
R001059	**User schema did not contain required value** *value* **for** *value*.
R001060	**Superior object class:** *object_class* **found during sups processing is obsolete.**
R001061	**The character '***character***' (***hex_character***) is not supported in substring search filters of attribute type:** *attr_type*.
R001062	**Substring search filters are not supported for attribute type:** *attr_type*.
R001063	**The keyword '***keyword***' is specified multiple times in value:** *value*
R001064	**'***date/time***' specifier is not valid for value:** *value*. **Valid values are** *values*.
R001065	**Attribute information not found for:** *attr_type*.
R001066	**User schema did not contain required value** *value* **for** *value*.
R001067	*required_value* **missing for schema value:** *value*.
R001068	**Adding attribute type:** *attr_type* **failed, because usage is not userApplications.**
R001069	**Reference attribute type not found when parsing IBM Attribute type:** *attr_type*.
R001070	**Schema entry did not contain all required attribute types.**
R001071	**Entry did not contain an object class.**
R001072	**More than one object class type keyword found in value:** *value*
R001073	**Empty quoted string found in value:** *value*.
R002001	**Missing equal sign in DN component:** *DN_component*.
R002002	**An escape sequence found in the following DN component is not valid:** *DN_component*.
R002003	**An internal server error has been encountered.**
R002004	**Missing closing quote, or incomplete escape sequence in the DN component:** *DN_component*.
R002005	**A character formed by an escape sequence in the following DN component is not valid UTF-8:** *DN_component*.
R002006	**Empty DN components are not supported.**
R002007	**Syntax of aclEntry attribute not as expected:** *value*.

R002008	Permissions missing for security level in: *value*.
R002009	Keyword: *keyword* not allowed multiple times in: *value*.
R002010	Search filter processing failed calling regcomp(), rc=*rc* (*"error_string"*), pattern=*"pattern"*.
R002011	Search filter processing failed calling regexec(), rc=*rc* (*"error_string"*), comparison value=*"value"*.
R002012	LDIF line missing separator (':') between attribute type and value: *value*.
R002013	LDIF line contains an attribute type (*attr_type*), but no value.
R002014	Incorrect base-64 encoded LDIF value found. Attribute type= *attr_type*. Bad character=*character*' (*hex_character*).
R002015	LDIF entry contains multiple DNs (first DN in entry= *DN*).
R002016	The following LDIF entry does not contain a DN: *LDIF-entry*.
R002017	The DN component *DN_component* contains an unescaped special character (*char*).
R002018	An extraneous colon was found in aclEntry value: *value*.
R002019	An unsupported extensible filter was detected.
R002020	AND filter encoding has no subfilter.
R002021	OR filter encoding has no subfilter.
R002022	NOT filter encoding has no subfilter.
R002023	NOT filter encoding has multiple subfilters.
R002024	SUBSTRING filter encoding has multiple initial strings.
R002025	SUBSTRING filter encoding has multiple final strings.
R002026	SUBSTRING filter encoding has unknown type (not INITIAL, ANY, FINAL).
R003001	ACLManager::checkAccess() is not implemented.
R003002	ACLManager::applyAdd() is not implemented.
R003003	ACLManager::applyAddForSrcEID() is not implemented.
R003004	ACLManager::applyAddForAclsrc EID() is not implemented.
R003005	ACLManager::applyAddForOwnsrc EID() is not implemented.
R003006	ACLManager::applyAddForPropagate() is not implemented.
R003007	ACLManager::applyAddModifyPre Scan() is not implemented.
R003008	ACLManager::applyAddModifyPost Apply() is not implemented.
R003009	ACLManager::applyAddModify() is not implemented.
R003010	ACLManager::applyAddDelete() is not implemented.
R003011	ACLManagerFactory::newACL Manager() is not implemented.
R003012	ACLManagerFactory::newACL Subject() is not implemented.
R003013	ACLManagerFactory::newACL Object(1) is not implemented.

R003014	**ACLManagerFactory::newACL Object(2) is not implemented.**
R003015	**ACLManagerFactory::newACL Object(3) is not implemented.**
R003016	**ACLManagerFactory::newACL Object(4) is not implemented.**
R003017	**ACLManagerFactory::newACL Object(5) is not implemented.**
R003018	**ACLManagerFactory::newACL Object(6) is not implemented.**
R003019	**ACLManagerFactory::newACLRights() is not implemented.**
R003020	**ACLManagerFactory::newACLMod Progress() is not implemented.**
R003021	**ACLManagerFactory::deleteACL Manager() is not implemented.**
R003022	**ACLManagerFactory::deleteACL Object() is not implemented.**
R003023	**ACLManagerFactory:: deleteACLSubject() is not implemented.**
R003024	**ACLManagerFactory::deleteACL Rights() is not implemented.**
R003025	**ACLManagerFactory::deleteACLMod Progress() is not implemented.**
R003026	**Failure in constructing the ACLManager instance, rc =** *rc*.
R003027	**Unable to determine parent EID during set of ACLSRC EID.**
R003028	**Unable to find first propagating EID during set of ACLSRC EID.**
R003029	**Either aclPropagate or aclEntry was missing from new entry.**
R003030	**Unable to determine parent EID during set of OWNSRC EID.**
R003031	**Unable to find first propagating EID during set of OWNSRC EID.**
R003032	**Either ownerPropagate or entryOwner was missing from new entry.**
R003033	**Incorrect value specified for aclPropagate, value =** *value*.
R003034	**Multiple values specified for aclPropagate.**
R003035	**Incorrect value specified for ownerPropagate, value =** *value*.
R003036	**Multiple values specified for ownerPropagate.**
R003037	**Incorrect aclPropagate setting found.**
R003038	**Incorrect aclPropagate setting found.**
R003039	**Either aclPropagate or aclEntry was missing from new entry during propagation processing.**
R003040	**Either ownerPropagate or entryOwner was missing from new entry during propagation processing.**
R003041	**ACLManagerImpl::applyModify PreScan() is not implemented.**
R003042	**ACLManagerImpl::applyModify PostApply() is not implemented.**
R003043	**ACLManagerImpl::applyModify() is not implemented.**
R003044	**Incorrect value =** *value* **found for aclprop.**
R003045	**Incorrect value =** *value* **found for ownprop.**

R003046	**Incorrect value specified for aclPropagate in modifications list, value =** *value*.
R003047	**Incorrect value specified for ownerPropagate in modifications list, value =** *value*.
R003048	**Unable to rebuild caches.**
R003049	**Unable to determine if subject is the Administrator.**
R003050	**Access allowed because subject is the Administrator.**
R003051	**Unable to build parent entry during check for add rights.**
R003052	**Unable to find parent EID and parent DN is not a suffix.**
R003053	**Unable to determine if subject is an Owner during add check.**
R003054	**Access allowed for add because subject is the owner of the entry.**
R003055	**Unable to find parent ACL during add check.**
R003056	**Access allowed because subject has add permission in parent ACL.**
R003057	**Access denied because subject does not have add permission in parent ACL.**
R003058	**Unable to find first propagating ACL EID during add check.**
R003059	**Unable to find effective ACL entry in first propagating ACL during add check.**
R003060	**Access denied for add because subject does not have an entry in propagating ACL.**
R003061	**Access allowed because subject has write permission to all attributes in new entry.**
R003062	**Access denied for add because subject does not have write permission to all attributes in new entry.**
R003063	**Incorrect return code =***rc* **encountered at end of access check.**
R003064	**Unknown ACLRIGHTS value encountered.**
R003065	**Unable to determine if subject is an Owner during modify check.**
R003066	**Access allowed for modify because subject is the owner of the entry.**
R003067	**Unable to find effective ACL entry during modify check.**
R003068	**Access denied for modify because subject does not have an entry in ACL.**
R003069	**Access allowed for modify because subject has write permission to all modified attributes.**
R003070	**Access denied for modify because subject does not have write permission to all modified attributes.**
R003071	**Unable to determine if subject is an Owner during delete check.**
R003072	**Access allowed for delete because subject is the owner of the entry.**
R003073	**Unable to find effective ACL entry in during delete check.**
R003074	**Access denied for delete because subject does not have an entry in ACL.**
R003075	**Access allowed for delete because subject has delete permission for the entry.**

R003076	**Access denied for delete because subject does not have delete permission on the entry.**
R003077	**Unable to determine if subject is an Owner during modify name check.**
R003078	**Access allowed for modify name because subject is the owner of the entry.**
R003079	**Unable to find effective ACL entry during modify name check.**
R003080	**Access denied for modify name because subject does not have an entry in ACL.**
R003081	**Access allowed for modify name because subject has write permission on all attributes in the name.**
R003082	**Access denied for modify name because subject does not have write permission on all attributes in the name.**
R003083	**Unable to determine if subject is an Owner during search check.**
R003084	**Access allowed for search because subject is the owner of the entry.**
R003085	**Unable to find effective ACL entry during search check.**
R003086	**Access denied for search because subject does not have an entry in ACL.**
R003087	**Access allowed for search because subject has read permission on all requested attributes.**
R003088	**Access allowed for search because subject has read permission some requested attributes.**
R003089	**Access denied for search because subject does not have read permission on all requested attributes.**
R003090	**Unable to determine if subject is an Owner during compare check.**
R003091	**Access allowed for compare because subject is the owner of the entry.**
R003092	**Unable to find effective ACL entry in during compare check.**
R003093	**Access denied for compare because subject does not have an entry in ACL.**
R003094	**Access allowed for compare because subject has compare permission on attribute.**
R003095	**Access denied for compare because subject does not have compare permission on attribute.**
R003096	**Access check indicated successful but allowed flag was not set.**
R003097	**Update of SRC column failed for case oldVal=NONE, newVal=TRUE.**
R003098	**Update of SRC column failed for case oldVal=NONE, newVal=FALSE.**
R003099	**Update of SRC column failed for case oldVal=TRUE, newVal=NONE.**
R003100	**Update of SRC column failed for case oldVal=TRUE, newVal=FALSE.**
R003101	**Update of SRC column failed for case oldVal=FALSE, newVal=NONE.**
R003102	**Update of SRC column failed for case oldVal=FALSE, newVal=TRUE.**
R003103	**Update results in aclPropagate value without an associated aclEntry value.**
R003104	**Update results in no aclPropagate value with an associated aclEntry value.**

R003105	Update results in ownerPropagate value without an associated entryOwner value.
R003106	Update results in no ownerPropagate value with an associated entryOwner value.
R003107	Unable to add ACL or owner information into entry.
R003108	ACLManager::isACLModified() is not implemented.
R003109	Unable to determine first propagating Owner for new entry.
R003110	Unable to determine if subject would be Owner for new entry.
R003111	ACLManager::getFirstPropagation AclEID() is not implemented.
R003112	ACLManager::getFirstPropogating OwnEID() is not implemented.
R003113	ACLManager Factory::newACLobject(7) is not implemented.
R003114	Access denied because newSuperior could not be determined.
R003115	Unable to determine if subject is an Owner during modify name check.
R003116	Unable to find effective ACL entry during modify name check.
R003117	Access denied for modify name (with newSuperior) because subject does not have an entry in ACL for target object.
R003118	Access allowed for modify name because subject has write permission on all attributes in the target object's name.
R003119	Access denied for modify name because subject does not have write permission on all atrributes in the name.
R003120	Unable to build parent entry during check for modify DN rights.
R003121	newSuperior designated as root and renamed DN is not a suffix (Modify DN).
R003122	Access allowed for modify name because subject has necessary permissions on both target and newSuperior objects.
R003123	Access denied for modify name because subject does not have add permission on newSuperior object.
R003124	Access denied for modify name (with newSuperior) because subject does not have delete permission on target object.
R003125	Access denied for modify name (with newSuperior) because subject does not have add permission on newSuperior object.
R003126	Access denied for modify name (with newSuperior) because subject does not have an entry in ACL for newSuperior object.
R003127	Error attempting to update entry with aclsource, ownersource, aclpropagate, and ownerpropagate information.
R004001	Unknown error occurred!
R004002	Container operation failed!
R004003	The base dn *DN* specified on the command is not valid.
R004004	Encrypt of directory entry password failed with rc =*rc*.
R004005	DN *DN* exceeds maximum length of *max_length*.
R004006	DN *DN* already exists.

R004007	**The parent DN *DN* specified is not valid.**
R004008	**Object *DN* does not exist.**
R004009	**Data Base utilities failed with rc = *rc*.**
R004010	**Get ancestors utilities failed with rc = *rc*.**
R004011	**SQL utilities failed with rc = *rc*.**
R004012	**Error decoding DN from database.**
R004013	**Error decoding attribute type from database for entry *DN*.**
R004014	**Error skipping the decoding of attribute type *attr_type* in entry *DN*.**
R004015	**Error decoding attribute values for attribute type *attr_type* in entry *DN*.**
R004016	**Error decoding long attribute values for attribute type *attr_type* in entry *DN*.**
R004017	**No attribute values found for entry *DN*.**
R004018	**Internal error encountered with the TDBMCacheManager.**
R004019	**Entry data is missing required RDN components.**
R004020	**Unexpected internal filter error.**
R004021	**RDN contains duplicate values for attribute *attr_type*, value = *value*.**
R004022	**No parent entry for *DN*.**
R004023	**Unable to create an ACL request object.**
R004024	**The filter used to search for the schema is not valid. Filter should be "objectclass=subschema".**
R004025	**Error in search checking to see if DN is a V2 referral.**
R004026	**Entry not found in the database.**
R004027	**LDAP Search is unwilling to perform the attempted search. The generated SQL statement is too large.**
R004028	**The size limit for your search has been reached.**
R004029	**Unable to send search results.**
R004030	**Search failed because it was unable to check ACLs for returned entries.**
R004031	**Time limit exceeded for the present search.**
R004032	**Attempt to scan referral cache failed.**
R004033	**Unrecognized filter type.**
R004034	**Unable to create entry structure for the entry object.**
R004035	**Attribute *attr_type* is not allowed to be modified by the user.**
R004036	***attr_type* attribute already contains value = value.**
R004037	**If the *attr_type* attribute is specified its value must be equal to the DN.**
R004038	**Operation not allowed. The configuration file specifies a read-only mode for this server.**
R004039	**Operation not allowed. The schema cannot be deleted from the server.**
R004040	**An error occurred checking the referral cache.**
R004041	**Entry *DN* is not a leaf. Deletion is not allowed.**

R004042	**Unable to determine if entry *DN* is a leaf. Deletion was unsuccessful.**
R004043	**Unable to create SQL to delete from the descendants table. Deletion of entry *DN* is unsuccessful.**
R004044	**Unable to delete entry *DN* from the descendants table. Deletion was unsuccessful.**
R004045	**Unable to create SQL to delete from the search table. Deletion of entry *DN* is unsuccessful.**
R004046	**Unable to delete entry *DN* from the search table. Deletion was unsuccessful.**
R004047	**Unable to create SQL to delete from the DIR_ENTRY table. Deletion of entry DN is unsuccessful.**
R004048	**Unable to delete entry *DN* from the DIR_ENTRY table. Deletion was unsuccessful.**
R004049	**Attempt to delete entry *DN* from replica servers failed. Deletion was unsuccessful.**
R004050	**Unable to create a request structure.**
R004051	**Entry *DN* does not contain attribute *attr_type*.**
R004052	**Entry size column is missing from select statement columns.**
R004053	**EID column is missing from select statement columns.**
R004054	**One or more characters found in string *string* are not valid UTF-8.**
R004055	**Attribute *attr_type* is not allowed to be modified by the user.**
R004056	**Modrdn of the schema entry is not allowed.**
R004057	**DBXGetData Failed.**
R004058	**Unable to get encryption method.**
R004059	**Credential cannot be encrypted.**
R004060	**Entry does not contain a password.**
R004061	**DBXExecDirect failed.**
R004062	**Credentials are not valid.**
R004063	**Unable to gather group information.**
R004064	**DBXBindParameter failed.**
R004065	**DBXFetch failed.**
R004066	**Unable to add/delete/replace values.**
R004067	**Authentication method is not SIMPLE.**
R004068	**Modifications were not specified.**
R004069	**Server unable to obtain a request structure.**
R004070	**Referral cache scan failed.**
R004071	**DN does not exist.**
R004072	**Unable to update the entry object.**
R004073	**Non-Leaf Referral test failed.**
R004074	**Unable to prepare for entry table updates.**
R004075	**Unable to remove old RDN values.**
R004076	**DNS style names are not allowed.**
R004077	**DN already exists.**
R004078	**Unable to prepare for search table updates.**

R004079	**Unable to perform database operations.**
R004080	**User canceled request.**
R004081	**Unable to update the entry table.**
R004082	**Replication updates failed.**
R004083	**New superior is not allowed.**
R004084	**Entry is not a leaf.**
R004085	**Out of memory.**
R004086	**Entry** *DN* **already contains attribute** *attr_type*, **value=***value*.
R004087	**Encryption of password failed.**
R004088	**Entry** *DN* **does not contain attribute** *attr_type*.
R004089	**Attribute values not specified for replace.**
R004090	**Non UTF-8 V3 data detected.**
R004091	**IA5 fence is on and non-IA5 V2 data is detected.**
R004092	**Error Converting V2 string from ISO8859-1 to UTF-8.**
R004093	**Error converting binary data to a textual value.**
R004094	**Attribute** *attr_type* **cannot be a component of the RDN.**
R004095	**Filter contained a NOT with an unrecognized attribute type, which is unsupported.**
R004096	**Entry** *DN* **does not contain attribute** *attr_type*, **value=***value.*
R004097	**Error decoding the entry from the database.**
R004098	**Filtering on non-textual attributes** *attr_type* **is not allowed.**
R004099	**Parent of new entry** *DN* **is referral entry** *DN*. **Operation not allowed.**
R004100	**TDBMBeCapability: Constructor missing a required initializer.**
R004101	**Constructor call to entry_attr_merge_becap failed.**
R004102	**TDBMBeCapability::operator= not implemented.**
R004103	**TDBMBeCapability::operator== not implemented.**
R004104	**TDBMBeCapability::default constructor not implemented.**
R004105	**TDBMBeCapability::getNextVal: Bad cursor argument.**
R004106	**TDBMBeCapability::getNextVal: NULL cursor argument.**
R004107	**The __passwd function failed; not loaded from a program controlled library.**
R004108	**TDBM backend __passwd API resulted in an internal error.**
R004109	**The password has expired.**
R004110	**The user id has been revoked.**
R004111	**The password is not correct.**
R004112	**A bind argument is not valid.**
R004113	**Native authentication cannot be performed when multiple UID values exist for entry** *DN*.

R004114	**Modify-delete of the old password for entry *DN* must occur before the modify-add of the new password for native authentication password updates.**
R004115	**More than one password cannot be specified for a native authentication password update on entry *DN*.**
R004116	**Password change not allowed for entry *DN*. The nativeUpdateAllowed configuration option has not been enabled.**
R004117	**Native authentication password replace is not allowed for entry *DN*.**
R004118	**Entry *DN* native user ID (ibm-nativeId,uid) is not defined to the Security Server.**
R004119	**Modify-add of the new password must occur after the modify-delete of the old password for native authentication password updates on entry *DN*.**
R004120	**Entry *DN* participates in native authentication so adding the userpassword attribute is not allowed.**
R004121	**Entry *DN* participates in native authentication but is not configured correctly. If useNativeAuth = ALL then the entry must contain the attribute ibm-nativeId or uid.**
R004122	**Unable to get cursor name for DELETE statement.**
R004123	**Allocation of statement handle failed.**
R004124	**DELETE at cursor statement execution failed.**
R004125	**Operation not supported. The schema entry cannot be used in the compare operation.**
R004126	**The Prepare of a SQL statement failed.**
R004127	**The Execute of a SQL statement failed.**
R004128	**Native authentication password change failed: \n. The new password is not valid, or does not meet requirements.**
R004129	**Unable to locate newSuperior *dn* in any backend.**
R004130	**Time limit exceeded for Modify DN operation - *routine-name*.**
R004131	**Determination of descendants to rename failed with rc = *return-code*.**
R004132	**Either the newSuperior DN must exist in this backend, or (if it does not) the new (renamed) DN MUST be a suffix in this backend.**
R004133	**The newSuperior DN is located in the subtree to be moved - not permitted.**
R004134	**Failure occurred while inserting new ancestor rows in hierarchy.**
R004135	**Failure occurred while deleting old ancestors rows from hierarchy.**
R004136	**Failure occurred while updating parent EID to newSuperior EID.**
R004137	**Failure occurred while updating subtree node levels.**
R004138	**Unable to locate effective ACLEntry for renamed entry.**
R004139	**Unable to locate effective ACLEntry for newSuperior entry.**
R004140	**Failure occurred while updating ACL and owner inheritance.**
R004141	**Modify DN operation requires new rdn containing non-zero length string.**
R004142	**Failure occurred while realigning DN attribute values to match renamed DNs with rc=*return-code*.**

R004143	Could not realign DN attributes because no attribute types were found with DN syntax.
R004144	Retrieval of newSuperior object failed, despite apparent presence in backend.
R004145	The newSuperior DN is not permitted to be a referral object.
R004146	Unable to complete DN realignment - generated SQL statement too large.
R004147	Cannot replicate complex Modify DN operation because a connection cannot be made with the replica. Modify DN operation is refused.
R004148	An attribute type which should be present in candidate for DN realignment was not found.
R004149	Unable to complete DN realignment - could not update entry object.
R004150	Unable to complete DN realignment - could not update DIR_SEARCH with changes.
R004151	Failure occurred while attempting to lock candidate rows in DIR_ENTRY for modify DN operation.
R004152	One or more replica servers lack support for complex Modify DN operations. Modify DN operation refused.
R004153	Parent of new entry *entry_dn* is an alias entry *parent_dn*. Operation not allowed.
R004154	Entry *entry_dn* is not a leaf. It cannot be modified to become an alias.
R004155	Alias entry *entry_dn* points to itself. Operation not allowed.
R004156	Alias entry *entry_dn* points to an entry that is a descendant of itself: *alias_dn*. Operation not allowed.
R004157	A loop was detected while dereferencing DN *entry_dn*.
R004158	Dereferencing DN *entry_dn* involved processing more RDNs than is supported (*RDN_limit*).
R004159	Dereferencing DN *entry_dn* failed because the resulting DN does not exist in this backend.
R004160	Entry *entry_dn* cannot be both an alias and a referral.
R004161	Entry *entry_dn* is an alias whose dereferenced dn *alias_dn* is at or below a referral entry. This is not supported.
R004162	Operation not allowed. The change log root entry cannot be deleted from the server.
R004163	NewSuperior DN is not permitted to be an alias object.
R004164	Changing a NOID in the minimum schema is not allowed (NOID=*noid*).
R004165	Error creating request structure - *error*.
R004166	Error converting *entry_dn* to a database entry.
R004167	Alias entry *entry_dn* points to schema entry. Operation not allowed.
R004168	Change log root entry must contain an explicit and propagating ACL and Owner. Operation not allowed.
R004169	Cannot locate attrID for NOID *noid*, possibly due to updates to schema entry. Retry the operation.
R004170	DN *dn* is not authorized for *function*.
R004171	Control *control* is not enabled.

R004172	*string* **is specified twice.**
R004173	**Dereference type not supported (***number***) for** *string***.**
R004174	**Internal error, expected normalized value is absent for attr** *attr_type***, value** *value***.**
R006001	**LDAP Client API** *api_name* **has returned an error code=***return_code* **with an error message =** *error_string***.**
R006002	**LDAP Client API** *api_name* **has returned an error code=***return_code* **with an error message =** *error_string***. Request will be retried.**
R006003	**A decoding error has been encountered while decoding attibute(s):** *attr_type***.**
R006004	**An encoding error has been encountered while encoding attibute(s):** *attr_type***.**
R006005	**A code page conversion error has been encountered.**
R006006	**An unsupported control with OID=***oid* **has been encountered.**
R006007	**Critical extension with OID=***oid* **not supported.**
R006008	**An LDAP Client API has returned an error code=***return_code* **with an error message=***error_string***. Request may be retried.**
R006009	**The extended operation request with OID=***oid* **requires the critical control with OID=***oid***.**
R006010	**The extended operation request with OID=***oid* **is not supported.**
R006011	**The extended operation request with OID=***oid* **does not support the critical control with OID=***oid***.**
R006012	**The extended operation has encountered a cache miss.**
R006013	**Mutex Exception. Errno=***errno***, Errno2=***errno2***, Error message =** *error_string***.**
R006014	**Null Pointer Exception occurred.**
R006015	**Reference Counting Exception occurred.**
R006016	**Change log not configured.**
R006017	**Change log not active.**
R006018	**Cannot decode** *field* **from request.**
R006019	**Incorrect value (***value***) for** *field***, parsed from request.**
R006020	**Incorrect value (***value***) for** *field***, parsed from request.**
R006021	**Unable to convert** *field* **(***value***) to DN, rc=*return_code*.**
R006022	**PC caller must be in supervisor state.**
R007000	**Bind version** *version* **not supported.**
R007001	**Only SIMPLE authentication method is supported for V2.**
R007002	**Only EXTERNAL, GSSAPI, CRAM-MD5, and DIGEST-MD5 mechanisms are supported for SASL authentication methods.**
R007003	**Only SIMPLE and SASL authentication methods are supported for V3.**
R007004	**Bind does not support manageDsaIT control (managedsait).**
R007005	**Server not configured for client authentication.**

R007006	**No client certificate dn available.**
R007007	**Server not configured for Kerberos GSSAPI bind.**
R007008	**Bind function not implemented for be=***backend***.**
R007009	**Error obtaining controls from ber.**
R007010	**Control verify failed.**
R007011	**Error on dn_normalize of dn** *dn***.**
R007012	**Backend get_groups routine for be=***backend* **failed.**
R007013	**GSSAPI bind crendentials are NULL.**
R007014	**Sasl bind should be in progress.**
R007015	**Credentials do not match those in the configuration file.**
R007016	**No backend selected for entry** *DN***.**
R007017	**Backend bind failed for entry** *DN***.**
R007018	**CRAM-DM5 protocol error on bind.**
R007019	**DIGEST-MD5 protocol error on bind.**
R007020	**CRAM-DM5 bind with native authentication turned on is not supported.**
R007021	**DIGEST-MD5 bind with native authentication turned on is not supported.**
R007022	**Error: Invalid CRAM-MD5 client response format.**
R007023	**Error: Could not locate uid attribute in schema to perform Mandatory Authentication bind.**
R007024	**Error: Found multiple entries with the same uid value** *uid* **on Mandatory Authentication bind.**
R007025	**Error: Authentication DN does not equal username DN in CRAM-MD5 bind.**
R007026	**Error: Mandatory Authentication bind not allowed to SDBM backend.**
R007027	**TLS is not supported on the connection.**
R007028	**TLS/SSL is active on the connection.**
R007029	**Other operations are outstanding for the connection.**
R007030	**Error: Multiple** *attribute-name* **attributes found in DIGEST-MD5 response.**
R007031	**Error: Required attribute** *attribute-name* **is missing in DIGEST-MD5 response.**
R007032	**Error: Quotations are missing around attribute** *attribute-name* **in DIGEST-MD5 response.**
R007033	**Error: Quotations are not needed around attribute** *attribute-name* **in DIGEST-MD5 response.**
R007034	**Error: Authorization DN does not equal username DN in DIGEST-MD5 bind.**
R007035	**Error: DIGEST-MD5 response length** *length* **is greater than 4096.**
R007036	**Error: DIGEST-MD5 response missing 'dn:' on 'authzid' attribute.**
R007037	**Error: DIGEST-MD5 response attribute** *attribute-name* **value is different than what was sent.**
R007038	**Credentials are not valid.**

R007039	**Referrals returned.**
R007040	**Size limit in bytes (*limit*) has been exceeded.**
R007041	**Error: The sizelimit of limit is less than the minimum size of *minimum_limit*.**
R007042	**Format of extended operation remote connect LDAP URL *url* is not valid.**
R007043	**Remote LDAP server *server* is down.**
R007044	**Error: ldap_unbind() of remote connection failed rc=*return_code*.**
R007045	**Error: ibmSizeLimitControl is not marked critical.**
R007046	**Error: getaddrinfo() failed host=*host*, rc=*return_code*, errstring=*string*.**
R007047	**Error: SASL EXTERNAL platform identity bind fails because SDBM backend is not configured.**
R007048	**Error: remoteConnectRequest extended operation is only supported on the PC interface.**
R007049	**Error: ldap_ssl_client_init() failed rc=*return_code*, gskReasonCode=*reason_code* on remote LDAP server server.**
R007050	**Error: gethostname() failed errno=*number*, errstring=*string*.**
R007051	**Error: DIGEST-MD5 response hostname or ipaddress in the digest-uri cannot be verified. Aborting DIGEST-MD5 authentication.**
R007052	**Error: server running in maintenance mode, operations restricted to peerServerDN or masterServerDN.**

Chapter 29. Performance tuning

Overview

Several server configuration options and facilities significantly affect the performance of the server. In addition, specific LDAP server database backends operate in conjunction with other products which may require tuning to accommodate the LDAP server. For example, the TDBM and GDBM database backends use DB2, which provides a large set of tuning options. The SDBM backend provides access to the RACF database, which has its own product specific tuning options. This chapter describes some of the things to consider when configuring your server for optimal performance.

General LDAP server performance considerations

Threads

The **commThreads** configuration option specifies the number of communication threads which handle requests from clients to the LDAP server. However, the primary role of each of these threads is to serve as a worker thread for processing client requests to the directory database.

Each communication thread is shared among client connections and is used to process requests as they occur. Therefore, this option does not need to be set nearly as large as the expected number of concurrently connected clients.

Each communication thread requires some resources of its own, including low storage, a connection to DB2 (when TDBM or GDBM are configured), and other system resources associated with threads. Therefore, you may want to avoid making this option larger than is needed.

It is recommended that **commThreads** be set to approximately two times the number of CPUs that are running in your LPAR. However, this is a general rule depending upon the activity that your LDAP server experiences.

If most requests are search requests retrieved from storage in TDBM caches or DB2 buffer pools, then additional **commThreads** might not provide much benefit. However, if most requests to the database require I/O wait time, then additional **commThreads** might allow more client requests to run concurrently.

Debug settings

Activating the LDAP server debug trace facility will impact performance. If optimal performance is desired, debug should only be activated when it is necessary to capture diagnostic information.

TDBM performance considerations

The z/OS LDAP server TDBM backend uses IBM Database 2™ (DB2), a powerful and scalable database product, for its data storage facility. In the most optimal LDAP environments, directory data is fairly static and the access for TDBM cached data is repetitive. In other environments, where directory data is updated frequently and the access for non-cached data is random, the power and scale of DB2 is used to enhance performance.

The following is included in this chapter:

- DB2 tuning to improve database access
- TDBM tuning which affects DB2 usage
- TDBM caching of data to avoid database read operations to DB2

DB2 tuning is important to ensure that TDBM requests which access the database in DB2 operate efficiently, and that response times do not increase as the database grows in size. Many general DB2 tuning guidelines are applicable to TDBM databases.

Also, there are choices in the initial setup of the TDBM database and in the TDBM backend section of the LDAP server configuration file which influence performance within DB2.

TDBM caches provide a significant benefit to performance, allowing the server to bypass read operations to the database. Optimizing the cache size is important to ensure a high percentage hit rate, without requiring excessive storage.

DB2 tuning

The following tasks relating to DB2 tuning are crucial to maintaining good performance. These tasks are typically performed by Database Administrators on most production DB2 data:

- Periodically reorganizing the TDBM database via the DB2 **reorg** utility
- Periodically maintaining the database statistics via the DB2 **runstats** utility
- Allocating DB2 buffer pools large enough to minimize I/O to the TDBM database

The TDBM table spaces and indexes should be reorganized periodically using the DB2 **reorg** utility. This helps to improve database access performance and to reclaim fragmented space.

Also, the **runstats** utility should be run periodically to gather and record information about the TDBM data in its table spaces, indexes, and partitions. This information is necessary for DB2 to select efficient access paths to the data by the LDAP server. This information is also useful to the Database Administrator for determining when the database should be reorganized. The recommended parameters to specify when using the **runstats** utility are shown below:

```
//SYSIN           DD *
   RUNSTATS TABLESPACE LDAPSRV.SEARCHTS REPORT YES
     TABLE(ALL)
     INDEX(LDAPSRV.DIR_SEARCHX1 KEYCARD
     FREQVAL NUMCOLS 1 COUNT 100
     FREQVAL NUMCOLS 2 COUNT 100)
   RUNSTATS TABLESPACE LDAPSRV.ENTRYTS REPORT YES
     TABLE ALL INDEX ALL KEYCARD
   RUNSTATS TABLESPACE LDAPSRV.DESCTS REPORT YES
     TABLE ALL INDEX ALL KEYCARD
   RUNSTATS TABLESPACE LDAPSRV.LENTRYTS REPORT YES
     TABLE ALL INDEX ALL KEYCARD
   RUNSTATS TABLESPACE LDAPSRV.LATTRTS REPORT YES
     TABLE ALL INDEX ALL KEYCARD
   RUNSTATS TABLESPACE LDAPSRV.MISCTS REPORT YES
     TABLE ALL INDEX ALL KEYCARD
   RUNSTATS TABLESPACE LDAPSRV.REPTS REPORT YES
     TABLE ALL INDEX ALL KEYCARD
 /*
```

Note: In the example above, **LDAPSRV** is the userid used to create the TDBM tables and indexes. This is the same value used in the LDAP server configuration file for **dbuserid**.

Many installations populate the z/OS LDAP directory with a large amount of initial data and then gradually grow the directory over time with routine updates and additions. In such cases, it is highly recommended that the **reorg** and **runstats** utilities be run immediately after the directory is populated with this initial data prior to roll out to production. If the initial population of data is done using an application (as opposed to the **ldif2tdbm** load utility provided with the z/OS LDAP server), it may be necessary to run **reorg** and **runstats** one or more times during the process of initially populating the directory. This may be needed to ensure DB2 uses efficient access paths based on the statistics gathered from the database, once it

contains a representative amount of information. Without this information, poor access paths may be chosen which cause increasing response times as the size of the database increases, and gradual slowing of the process of populating the directory.

DB2 buffer pool allocations should also be examined to ensure they are sufficient for the LDAP TDBM database. It is often useful to isolate specific TDBM table spaces and indexes to their own buffer pools. In particular, separating the indexes from the table spaces may help ensure that the index buffers remain in the buffer pools. This technique may also help evaluate overall behavior of the LDAP database regarding its buffer pool usage when specific tables and indexes correlate to specific buffer pools. Products such as the DB2 Performance monitor are especially useful for monitoring buffer pool activity.

TDBM database tuning

Several choices may ultimately affect the performance of TDBM when accessing its data within DB2:

- The size of the **DN_TRUNC** column of the **DIR_ENTRY** table specified at database creation time.

  The **DN_TRUNC** column is used to index data in the **DIR_ENTRY** table to speed up retrieval of directory entries via their distinguished name (DN). This column holds the leading portion of each DN, and should be defined long enough to make most values unique.

  Some applications generate directory entries where the leading portion of the DN is identical. For example, Tivoli Access Manager (TAM) generates entries under each user entry in the namespace where the DN starts with ″**cn=secPolicyData,secAuthority=Default,**″. To provide uniqueness, it is recommended that installations using TAM with the z/OS LDAP server, define the **DN_TRUNC** column to be 64 bytes in length.

  You should define this column at its proper length during initial set up of the directory. Changing the size requires the **DIR_ENTRY** table to be redefined, and thus the directory must be unloaded and reloaded to implement the change.

- The size of the **VALUE** column of the **DIR_SEARCH** table specified at database creation time.

  The **VALUE** column is used to index data in the **DIR_SEARCH** table to speed up retrieval of directory entries for search requests using the search filter values. This column holds the leading portion of textual attribute values, and should be defined long enough to accommodate most values specified in search filters. However, this column should not be made significantly larger than required, since this may cause the **DIR_SEARCH** table and its index to substantially increase in size.

  You should define this column at its proper length during initial set up of the directory. Changing the size requires the **DIR_SEARCH** table to be redefined, and thus the directory must be unloaded and reloaded to implement the change.

- The **attrOverflow** size specified in the TDBM section of the LDAP server configuration file.

  This configuration option specifies the threshold size of attribute values which are stored separately from the **DIR_ENTRY** data and are instead stored in the **DIR_LONGATTR** overflow table.

  This option can avoid unnecessarily reading this overflow data for searches which do not request the attribute. For example, if your directory entries contain JPEG data, but many searches ask for specific attributes and omit the large JPEG attribute from those requested, this option can help avoid reading unnecessary data from the database.

  This option value should be specified large enough so that data which is typically retrieved with the entry remains in the **DIR_ENTRY** data. Note that entries with overflow data are not eligible for the entry cache, so making this option value too small can impact search performance.

TDBM cache tuning

The TDBM backend implements many caches to help reduce processing time and to avoid access to the DB2 database. These caches are beneficial when most accesses to the directory are read operations. Tuning these caches involves monitoring their effectiveness and adjusting their size to increase the percent hit rate.

Note: Increasing cache sizes may increase the amount of storage required by the server.

Some caches are invalidated by update activities. If this is a frequent occurrence, increasing the cache size may be of little or no benefit. If the cache hit rate is never any higher than zero for a particular cache, the cache can be disabled by setting its size to 0. However, even caches with seemingly low cache hit rates might provide some benefit, so one should generally avoid disabling them unless close monitoring is done to ensure they are not beneficial.

All implemented caches in TDBM are enabled by default, and the default sizes will generally provide some benefit to most installations. However, many installations might benefit from additional tuning. The following approach can be used to evaluate the cache sizes:

- Monitor the cache performance during typical workloads:

  You can use either the **cn=monitor** search or the operator console **MODIFY** command to retrieve current cache statistics. The **cn=monitor** search is described later in this chapter. For details on the operator console **MODIFY** command, see the description of the MONITOR report in "Capturing performance information" on page 109.

  Note: The monitor search must be used with a scope of subtree or one-level to retrieve the cache statistics, since the caches are backend specific.
- Examine the cache hit rate, the current number of entries, and the maximum allowed entries (configured size). Also, note the number of cache refreshes and the average size of the cache at refresh.
- If the cache hit rate is well below 100% and the cache is frequently fully populated, consider increasing the cache size. Since this is a configuration option, you must change the server configuration file and restart the server to affect the change.

The following caches are implemented in TDBM:

ACL source cache
: This cache holds information regarding ACL definitions within the database. Retrieval of information from this cache avoids database read operations when resolving access permissions.

DN cache
: This cache holds information related to the mapping of distinguished names between their raw form and their canonical form. Retrieval of information from this cache reduces processing required to locate entries in the database.

DN to eid cache
: This cache holds information related to the mapping of distinguished names in their canonical form and their entry identifier within the database. Retrieval of information from this cache avoids database read operations when locating entries within the database.

entry cache
: This cache holds information contained within individual entries in the database. Retrieval of information from this cache avoids database read operations when processing entries within the database.

entry owner cache
: This cache holds information regarding ACL definitions within the database. Retrieval of information from this cache avoids database read operations when resolving access permissions.

filter cache
: This cache holds information related to the mapping of search request inputs and the result set. Retrieval of information from this cache avoids database read operations when processing search requests.

Monitoring performance with cn=monitor search

You can retrieve statistics from the server by issuing a search request with a search base of **cn=monitor** and a filter of (**objectclass=***). These are the only values accepted for search base and filter on the monitor search. However, any of the possible scope values are accepted.

The z/OS LDAP server presents monitor data back in multiple entries. Server-wide statistics are returned in an entry whose distinguished name is **cn=monitor**. Also, each configured backend has statistics returned in its own entry named **cn=backend***XXXX***,cn=monitor**, where *XXXX* is the backend name specified on the database line in the server configuration file. If no backend name is specified, one is generated. In addition, the naming contexts pertaining to the specific backend are included in the output to identify which server backend is being reported.

For a scope of:

base only the **cn=monitor** entry is returned containing server-wide statistics

one (one-level search)
all backend-specific entries are returned

sub (subtree search)
all entries are returned

Statistics generally reflect data gathered since the LDAP server was started. However, many of the counters can be reset by using the **MODIFY** *ldapsrv***,APPL=QUERY,MONITOR,RESET** console command, where *ldapsrv* is the LDAP server jobname. In this case, the values reflect data gathered since the last reset.

The monitor search returns the following attributes:

Table 48. Server statistics

version	Version of the LDAP server
livethreads	Configured number of communication threads (commThreads)
maxconnections	Configured maximum number of connections (maxConnections)
sysmaxconnections	System defined maximum number of connections
totalconnections	Number of client connections made to the server
currentconnections	Current number of client connections
maxreachedconnections	High water mark for concurrent client connections
currenttime	Current date and time on the server
starttime	Date and time the server was started
resettime	Date and time statistics were last reset
resets	Number of times statistics were reset

The **sysmaxconnections** value may be lower than the **maxconnections** value because of system limits. If the value for the **maxConnections** configuration option is not valid, the **maxconnections** attribute value on **cn=monitor** search will reflect the system maximum connection limit. For information on how the maximum number of client connections is set in the LDAP server, refer to the **maxConnections** configuration option on page 68. When statistics are reset, **resettime** is set to the value of **currenttime**, **resets** is incremented, and **maxreachedconnections** is set to the value of **currentconnections**. None of the other **Server Statistics** listed above are affected by a reset.

Table 49. Server and backend specific statistics

opsinitiated	Number of operations initiated
opscompleted	Number of operations completed
abandonsrequested	Number of abandon operations requested
abandonscompleted	Number of abandon operations completed
addsrequested	Number of add operations requested
addscompleted	Number of add operations completed
bindsrequested	Number of bind operations requested
bindscompleted	Number of bind operations completed
comparesrequested	Number of compare operations requested
comparescompleted	Number of compare operations completed
deletesrequested	Number of delete operations requested

Table 49. Server and backend specific statistics (continued)

deletescomplketed	Number of delete operations completed
extopsrequested	Number of extended operations requested
extopscompleted	Number of extended operations completed
modifiesrequested	Number of modify operations requested
modifiescompleted	Number of modify operations completed
modifydnsrequested	Number of modifyDn operations requested
modifydnscompleted	Number of modifyDn operations completed
searchesrequested	Number of search operations requested
searchescompleted	Number of search operations completed
unbindsrequested	Number of unbind operations requested
unbindscompleted	Number of unbind operations completed
unknownopsrequested	Number of unrecognized operations completed
unknownopscompleted	Number of unrecognized operations completed
bytessent	Number of bytes of data sent
entriessent	Number of search entries sent
searchreferencessent	Number of search references sent

When statistics are reset, all of the **Server and Backend Specific Statistics** listed above are set to zero.

Table 50. Backend specific statistics

acl_source_cache_size	Configured maximum size (in entries) of the ACL Source cache (aclSourceCacheSize)
acl_source_cache_current	Current size (in entries) of the ACL Source cache
acl_source_cache_hit	Number of lookups that have hit the ACL Source cache
acl_source_cache_miss	Number of lookups that have missed the ACL Source cache
acl_source_cache_percent_hit	Percent of lookups that have hit the ACL Source cache
acl_source_cache_refresh	Number of times the ACL Source cache was invalidated
acl_source_cache_refresh_avgsize	Average number of entries in the ACL Source cache at invalidation
dn_cache_size	Configured maximum size (in entries) of the DN cache (dnCacheSize)
dn_cache_current	Current size (in entries) of the DN cache
dn_cache_hit	Number of lookups that have hit the DN cache
dn_cache_miss	Number of lookups that have missed the DN cache
dn_cache_percent_hit	Percent of lookups that have hit the DN cache
dn_cache_refresh	Number of times the DN cache was invalidated
dn_cache_refresh_avgsize	Average number of entries in the DN cache at invalidation
dn_to_eid_cache_size	Configured maximum size (in entries) of the DN to Entry ID cache (dnToEidCacheSize)
dn_to_eid_cache_current	Current size (in entries) of the DN to Entry ID cache
dn_to_eid_cache_hit	Number of lookups that have hit the DN to Entry ID cache
dn_to_eid_cache_miss	Number of lookups that have missed the DN to Entry ID cache
dn_to_eid_cache_percent_hit	Percent of lookups that have hit the DN to Entry ID cache
dn_to_eid_cache_refresh	Number of times the DN to Entry ID cache was invalidated
dn_to_eid_cache_refresh_avgsize	Average number of entries in the DN to Entry ID cache at invalidation
entry_cache_size	Configured maximum size (in entries) of the Entry cache (entryCacheSize)
entry_cache_current	Current size (in entries) of the Entry cache
entry_cache_hit	Number of lookups that have hit the Entry cache
entry_cache_miss	Number of lookups that have missed the Entry cache
entry_cache_percent_hit	Percent of lookups that have hit the Entry cache
entry_cache_refresh	Number of times the Entry cache was invalidated
entry_cache_refresh_avgsize	Average number of entries in the Entry cache at invalidation

Table 50. Backend specific statistics (continued)

entry_owner_cache_size	Configured maximum size (in entries) of the Entry Owner cache (entryOwnerCacheSize)
entry_owner_cache_current	Current size (in entries) of the Entry Owner cache
entry_owner_cache_hit	Number of lookups that have hit the Entry Owner cache
entry_owner_cache_miss	Number of lookups that have missed the Entry Owner cache
entry_owner_cache_percent_hit	Percent of lookups that have hit the Entry Owner cache
entry_owner_cache_refresh	Number of times the Entry Owner cache was invalidated
entry_owner_cache_refresh_avgsize	Average number of entries in the Entry Owner cache at invalidation
filter_cache_size	Configured maximum size (in entries) of the Filter cache (filterCacheSize)
filter_cache_current	Current size (in entries) of the Filter cache
filter_cache_hit	Number of lookups that have hit the Filter cache
filter_cache_miss	Percent of lookups that have hit the Filter cache
filter_cache_percent_hit	Percent of lookups that have hit the Filter cache
filter_cache_refresh	Number of times the Filter cache was invalidated
filter_cache_refresh_avgsize	Average number of entries in the Filter cache at invalidation
filter_cache_bypass_limit	Configured Filter cache bypass limit (filterCacheBypassLimit)

When statistics are reset, the **cache_hit, cache_miss, cache_percent_hit, cache_refresh**, and **cache_refresh_avgsize** for each cache are reset to zero. Resetting the statistics has no effect on the **cache_size** for each cache, nor on the **filter_cache_bypass_limit**, since these are configured values. Resetting the statistics also has no effect on the **cache_current** for each cache, since the contents of the caches are not altered by a reset of statistics. Some caches may get invalidated and refreshed due to directory update operations. When this occurs, **cache_refresh** is incremented and **cache_current** is set to zero to reflect the refreshed (empty) cache. The **cache_hit, cache_miss**, and values **cache_percent_hit** are accumulated across cache invalidation and refresh.

Monitor search examples

Following is an example of a monitor search using scope=base. This returns only statistics related to the entire server:

```
ldapsearch -h ldaphost -p ldapport -b cn=monitor -s base objectclass=*
cn=monitor
version=z/OS Version 1 Release 6 Integrated Security Services LDAP Server
livethreads=10
maxconnections=1000
sysmaxconnections=500
totalconnections=101
currentconnections=1
maxreachedconnections=32
opsinitiated=2497891
opscompleted=2497890
abandonsrequested=0
abandonscompleted=0
addsrequested=5726
addscompleted=5726
bindsrequested=100
bindscompleted=100
comparesrequested=352606
comparescompleted=352606
deletesrequested=5716
deletescompleted=5716
extopsrequested=0
extopscompleted=0
modifiesrequested=12614
modifiescompleted=12614
modifydnsrequested=0
modifydnscompleted=0
```

```
searchesrequested=2121029
searchescompleted=2121028
unbindsrequested=100
unbindscompleted=100
unknownopsrequested=0
unknownopscompleted=0
entriessent=615893
bytessent=222490994
searchreferencessent=0
currenttime=Sat Mar 13 02:43:20 2004
starttime=Fri Mar 12 18:31:17 2004
resettime=Fri Mar 12 18:31:17 2004
resets=0
```

Following is an example of output of a monitor search with scope=one for a server using a single TDBM backend database. This returns only backend specific statistics. The cache statistics shown would only be included for TDBM and GDBM backends, since the other backend types do not implement caches:

```
ldapsearch -h ldaphost -p ldapport -b cn=monitor -s one objectclass=*
```

```
cn=backendtdbm1,cn=monitor
namingcontexts=O=IBM.COM
namingcontexts=SECAUTHORITY=DEFAULT
namingcontexts=CN=EXTRA_SUFFIX
opsinitiated=2497640
opscompleted=2497640
abandonsrequested=0
abandonscompleted=0
addsrequested=5726
addscompleted=5726
bindsrequested=0
bindscompleted=0
comparesrequested=352606
comparescompleted=352606
deletesrequested=5716
deletescompleted=5716
extopsrequested=0
extopscompleted=0
modifiesrequested=12614
modifiescompleted=12614
modifydnsrequested=0
modifydnscompleted=0
searchesrequested=2120978
searchescompleted=2120978
unbindsrequested=0
unbindscompleted=0
unknownopsrequested=0
unknownopscompleted=0
entriessent=615750
bytessent=222292715
searchreferencessent=0
acl_source_cache_size=100
acl_source_cache_current=99
acl_source_cache_hit=611502
acl_source_cache_miss=331923
acl_source_cache_percent_hit=64.82%
acl_source_cache_refresh=0
acl_source_cache_refresh_avgsize=0
dn_cache_size=1000
dn_cache_current=986
dn_cache_hit=38287567
dn_cache_miss=25204086
dn_cache_percent_hit=60.30%
dn_cache_refresh=0
dn_cache_refresh_avgsize=0
dn_to_eid_cache_size=1000
dn_to_eid_cache_current=234
dn_to_eid_cache_hit=2137922
```

```
dn_to_eid_cache_miss=14768
dn_to_eid_cache_percent_hit=99.31%
dn_to_eid_cache_refresh=0
dn_to_eid_cache_refresh_avgsize=0
entry_cache_size=5000
entry_cache_current=4983
entry_cache_hit=89097
entry_cache_miss=30814
entry_cache_percent_hit=74.30%
entry_cache_refresh=0
entry_cache_refresh_avgsize=0
entry_owner_cache_size=100
entry_owner_cache_current=2
entry_owner_cache_hit=996602
entry_owner_cache_miss=3
entry_owner_cache_percent_hit=100.00%
entry_owner_cache_refresh=0
entry_owner_cache_refresh_avgsize=0
filter_cache_size=5000
filter_cache_current=4991
filter_cache_hit=544303
filter_cache_miss=1571716
filter_cache_percent_hit=25.72%
filter_cache_refresh=22597
filter_cache_refresh_avgsize=3
filter_cache_bypass_limit=100

cn=backendcdbm1,cn=monitor
namingcontexts=CN=MONITOR
opsinitiated=50
opscompleted=49
abandonsrequested=0
abandonscompleted=0
addsrequested=0
addscompleted=0
bindsrequested=0
bindscompleted=0
comparesrequested=0
comparescompleted=0
deletesrequested=0
deletescompleted=0
extopsrequested=0
extopscompleted=0
modifiesrequested=0
modifiescompleted=0
modifydnsrequested=0
modifydnscompleted=0
searchesrequested=50
searchescompleted=49
unbindsrequested=0
unbindscompleted=0
unknownopsrequested=0
unknownopscompleted=0
entriessent=145
bytessent=200233
searchreferencessent=0
```

For information about monitoring performance with the modify operator command, see “Capturing performance information” on page 109.

Large access groups considerations

Users with large access groups in z/OS LDAP may experience performance problems and increased storage usage in the LDAP server as access groups grow in size. Tivoli Access Manager (TAM) users are susceptible to this.

TAM users often create access groups in LDAP containing many members with every user in the registry defined in one large access group. The performance impacts may worsen as the registry grows, with any of the following symptoms :

- Increased response time in the application, in both administrative tasks (such as adding users to the Registry) and user tasks (such as authenticating through TAM)
- Increased processor utilization in the LDAP server
- Increased storage requirements in the LDAP server
- Increased resource consumption in DB2 (logging, I/O, processor usage, buffer pool demands)

Customers experiencing these symptoms should ensure that APAR OA07189 is installed. The enhancements provided in APAR OA07189 improve performance for the typical TAM activities associated with large access groups, such as:

- Adding new members to the access group using LDAP modify
- Removing members from the access group using LDAP modify
- Determining if a member is in the group using LDAP compare

However, even after installing APAR OA07189, this does not mean access groups can become arbitrarily large. Some scenarios still require substantial amounts of processing and storage within the z/OS LDAP server, such as:

- A search operation which returns all the members of a large access group. This includes either a search which returns the many values with the member attribute, or a search which returns the many values in the **ibm-allmembers** operational attribute.
- A search operation which requests all the members of a large access group, but the members are not returned because ACL read permissions prevent the requester from seeing the data.
- Update requests which touch a large access group entry when **persistentSearch=yes** is configured for the TDBM backend which contains the large entry.

The addressability limits of the z/OS LDAP server may become a factor when there are hundreds of thousands or millions of members in a single access group.

These scenarios are also susceptible to the affects of LE HEAPPOOL usage as described below.

If APAR OA07189 is applied and the addressability limits of the LDAP server continue to be a factor due to the presence of large access groups in the directory, consider the following corrective actions:

- Increase the LDAP server's region size, if possible.
- Limit the number of members placed within a single access group and partition the users into separate access groups. The number of members for each access group which can be managed successfully will depend on many factors, such as the size of the member values, the amount of region defined for the z/OS LDAP server, and the level of concurrent activity within the server.
- Ensure that **persistentSearch=yes** is not configured unless it is required. Specifically, this pertains to the TDBM backend which contains the large access groups. Some applications which exploit **persistentSearch** may only need this function with the changelog in the GDBM backend. In this case, specifying **persistentSearch=yes** in the GDBM backend and **persistentSearch=no** in the TDBM backend will be enough, and will avoid reading the entire large group entries in the TDBM backend on update requests.

LE HEAPPOOLS considerations

By default, the z/OS LDAP server uses LE HEAPPOOLS to improve performance. This facility reduces the processor consumption and allows better parallelism of concurrent requests within the z/OS LDAP server. However, overall storage consumption is typically larger with the use of LE HEAPPOOLs as compared to running without the facility enabled. Also, once storage is allocated to a given LE HEAPPOOL, it remains allocated to that HEAPPOOL and can only be used for future storage requests that are eligible (based on

size) for the given HEAPPOOL. In some cases, when the z/OS LDAP server must process the entire large access group entry in storage, the following may occur:

- While the request is processing, the z/OS LDAP server may use all available storage in its address space, causing a failure of the request, a failure of other concurrent requests, or a failure and abnormal termination of the server.
- Due to the sudden, large demand for storage to process the large group, most or all of the storage available to the z/OS LDAP server may be allocated and reserved to specific HEAPPOOLs. Although the z/OS LDAP server may appear to be available and able to process a variety of requests, many subsequent requests may fail due to insufficient storage, particularly those for entries with large or numerous attributes. In the absence of any failures, this large increase in storage use by the z/OS LDAP server may be detectable by system resource monitoring products, such as the Resource Measurement Facility (RMF).

If these problems occur, consider either tuning the HEAPPOOL sizes or disabling the HEAPPOOL for the z/OS LDAP server.

Tuning the HEAPPOOL sizes optimizes storage usage for the data within the LDAP server. See *z/OS Language Environment Programming Guide* for details on how to tune the HEAPPOOL settings. Note that the procedure for tuning HEAPPOOL settings requires a controlled environment with representative workloads. In this case, the workload should include the scenarios described earlier which cause the large demands for storage. Note that it is recommended that the storage reports needed for the tuning procedure be gathered in a non-production environment because tracking the storage statistics significantly impacts performance.

Disabling HEAPPOOLs reduces the total heap storage requirements of the LDAP server, at the cost of increased processing.

Overriding the HEAPPOOL settings can be done by including 'HEAPPOOLS' specifications in the PARM field on the EXEC statement when running the LDAP server as a started task or job, or in the setting of the _CEE_RUNOPTS environment variable within the USS shell environment. For more details on setting this parameter, see *z/OS Language Environment Programming Reference* and *z/OS Language Environment Programming Guide*.

GDBM (Changelog) performance considerations

The GDBM database is used only for the changelog function. By its very nature, this function tends to have a high intensity of update activity compared to read activity. Since update activity is generally more costly than read activity, this function should only be enabled when its use is actually needed.

Since the GDBM database is modeled closely after the TDBM database implementation, many of the performance considerations are identical. However, the following differences should be noted:

- The distinguished names (DNs) of entries and the searchable attributes within entries in GDBM tend to be well bounded in size and content. As such, the default sizes for the **DN_TRUNC** column in the **DIR_ENTRY** table and the **VALUE** column in the **DIR_SEARCH** table do not require adjustment.
- Since most GDBM requests are update operations, the search filter cache and entry cache are disabled by default. You may enable these caches, if desired, but if this is done, it is recommended that these caches be monitored to ensure they are providing any benefit.
- When options **changeLogMaxAge** or **changeLogMaxEntries** are on, changelog is periodically trimmed, based on the limits set in the configuration file. For more information about these configuration options, see "Configuration file options" on page 56.

SDBM performance considerations

The z/OS LDAP server SDBM backend allows access to the RACF database. Most tuning which affects performance in this area is within the RACF product. Refer to *z/OS MVS Initialization and Tuning Guide* for more information about tuning RACF.

Also, see “SDBM operational behavior” on page 220 for details regarding different types of LDAP requests supported, and the RACF operations issued by these requests. This may also be helpful when assessing RACF tuning considerations.

When writing applications which only require authentication to the SDBM backend via LDAP bind requests, performance can be improved by specifying the **authenticateOnly** control on the bind request within the application. See “authenticateOnly” on page 459 for more information.

Chapter 30. Debugging the LDAP server

This chapter provides specific information to help you debug problems encountered using the z/OS LDAP server.

The LDAP server provides the following methods for reporting problems:

- **Server messages:**

  The LDAP server typically issues error messages to indicate a problem initializing the LDAP server or a problem processing a client request. An example of an error message is:

  ```
  GLD0027E slapd unable to start because all backends failed to configure.
  ```

  Informational messages supply non-error information on LDAP server configuration and request processing. An example is:

  ```
  GLD0190I Secure communications is active for IP: INADDR_ANY, secure port 636.
  ```

  All server messages are written to the job spool (the SLAPDOUT DD) of the LDAP server started task. Some critical messages are also written to the system console.

- **Server debug records:**

  The LDAP server can produce debug records when debug is set on. See Chapter 9, “Running the LDAP server,” on page 101 for further details on how to turn debug on and the various levels of debug records that can be produced. Note that turning debug on does impact LDAP server performance and should only be used when necessary.

  Each line of debug output is prefixed with a thread number and a timestamp. The thread number identifies the execution thread to which the debug record event applies. The remaining part of the debug record is composed of text or data values. Typically, these records indicate flow (routine entry/exit), key events, key data being processed, exceptions, and error conditions. Below is an example of debug output:

  ```
  ( 0x00000001 Tue Apr  6 15:47:48 2004 958378 ): => finish_read_config: configuring tdbm
  ( 0x00000001 Tue Apr  6 15:47:48 2004 958545 ): finish_read_config: no suffixes configured!
  ```

  The LDAP server provides two places to which debug output can be written:

  1. By default, server debug records are written to STDERR, which is directed to the SLAPDOUT DD of the LDAP server started task. These trace records are only produced when requested upon server startup with the **-d** option or with the console **modify** command once the server is running.
  2. Debug records can be written to an internal wrap trace table in the LDAP server. This table can be dynamically printed into a dataset upon request. See “Gathering trace records into the in-storage trace table” on page 107 for more information.

  For problems that are quickly or easily recreatable, you can run with debug levels All-STRBUF (2146959359). For non-recreatable problems, you should limit the levels of debug records and use the internal wrap trace table.

- **Return and reason codes:**

  When the LDAP server encounters an error processing a client request, the server returns a return code and an LDAPResult structure containing information about the error. The information includes:

  – a numeric reason code
  – textual diagnostic information
  – traceback information indicating what code generated the error

  The return code is provided at +x’8’ into the return buffer, just before the LDAPResult structure.

  Below is an example of the results buffer returned by the LDAP server to the client. Take note of the x’13’ return code at x’8’ into the buffer, followed by the LDAPResult structure containing the reason code (R001028), the text writeup for the reason code, and the traceback information.

```
OSet  Address = 26F3E363 Length = 00CA  ASCII                EBCDIC
0000  02010769 81C40A01 13040004 81BC5230  ...i.........R001028  ....aD......a...
0010  30313032 38204D6F 72652074 68616E20  More than one value  ......(?...../>.
0020  6F6E6520 76616C75 65207370 65636966  specified for single  ?>.../%.........
0030  69656420 666F7220 73696E67 6C652076  valued attribute  .....?....>.%...
0040  616C7565 64206174 74726962 75746520  type: ibm-wsnname.  /%..../.........
0050  74797065 3A206962 6D2D7773 6E6E616D  (schemaimpl.c1.77|1900),  .`......_...>>/_
0060  652E2028 73636865 6D61696D 706C2E63  (schemaimpl.c|1.771906),  ........_/._.%..
0070  7C312E37 377C3139 3030292C 20287363  (schemaimpl.c|1.77|1914),  @....@.........
0080  68656D61 696D706C 2E637C31 2E37377C  (tdbm_add.c|1.56.1.2|699)  .._/._.%..@....@
0090  31393036 292C2028 73636865 6D61696D                          ...........__/._
00A0  706C2E63 7C312E37 377C3139 3134292C                          .%..@....@......
00B0  20287464 626D5F61 64642E63 7C312E35                          .....__/....@...
00C0  362E312E 327C3639 3929                                       .....@....
```

Classifying debug problems

Failures with the LDAP server can be classified as:

- server startup failures
- failures processing client requests after the server is up and running

Server startup problems

How to debug

When the z/OS LDAP server fails to properly initialize, you will receive one or more error messages. Some messages provide sufficient text to be able to resolve the problem. For example:

```
GLD3017I Unable to connect to the database, rc=1.
```

In this scenario, the problem was that the DB2 database was not available to connect to. There are some cases where the message(s) provided from a server that does not start are not sufficient to determine the cause of the problem. For example:

```
GDL0027E slapd unable to start because all backends failed to configure.
```

To help determine the cause of the LDAP server not being able to successfully initialize, you will need to gather a debug trace. You can request this by using one of the methods below:

1. On the start command, add the debug parameter. For example:

   ```
   s ldapsrv -d 65535
   ```

2. Pass the -d option value on the parms option in the JCL:

   ```
   // PARMS='-d 65535'
   ```

Note: In the examples above, the debug level of 65535 is used. This level of debug will gather all debug levels from 'trace' up through 'error'. For more information about LDAP server debugging, see Chapter 9, "Running the LDAP server," on page 101.

Common problems for server startup

Below are some common problems experienced when bringing up the LDAP server:

1. **Configuration file problems**: Also see "Configuration file format" on page 53 for more information.

   sequence numbers at ends of lines
   may cause various error messages to occur, depending on what configuration options are involved. Do not use sequence numbers in your active LDAP configuration file.

   entries not starting in column 1
   may also cause various error messages to occur, depending on what configuration options are

involved. This is because a line that begins with one or more blank spaces is processed as a continuation of the previous line. For example, if a backend has only one **suffix** option and that option does not start in column 1, then that backend will not initialize.

database option outside of the database section
are considered invalid options and are ignored, typically causing the database section appearing later in the configuration file to be incomplete and a particular configured backend may not initialize.

global options in the database section
may be ignored. Various errors will occur based on what global option was ignored.

no backends configured
Be sure you properly configure at least one backend.

overlapping suffixes between backends
If you overlap TDBM suffixes, you will receive error message GLD3040A. The SDBM backend can only have one suffix, so you cannot overlap SDBM suffixes. If you overlap suffixes between two different types of backends (TDBM and SDBM), this does not produce an error message and incorrect results may occur.

do not use (by uncommenting) options in the prefix commentary
(listed at the top of the configuration file) as this can cause various problems. A typical problem occurs when a database option is uncommented in this prefix section, which puts this live option in the global section instead of the database section. Do not uncomment any option in this prefix section as they are meant to be illustrative only.

more than one suffix option for the sdbm database section
causes GLD01041I

GLD0050E Attempt to bind failed, errno 111 EACCESS
occurs when the port you have specified is not available for use. Try another port.

GLD3091A An internal exception occurred, rc = *return_code***. Exception text is:** *exception_text***.**
may occur with reason code R004001, which results from having an invalid **dbuserid** option in the configuration file.

GLD0142E The value *URL_value* **for the** *configuration_option* **is not a valid URL or cannot be resolved.**
may occur when you use a *hostname* on the **listen** option in the configuration file or command line, and that *hostname* cannot be resolved. Try using *IP_address* on the **listen** option or ensure that resolution of the *hostname* is correct.

2. **DB2 Setup problems**:

SQL Native return code -924 / SQL State 58006
Prior to z/OS Version 1 Release 6, this symptom occurred when DB2 became unavailable to the LDAP server. The existing connections were dropped. In z/OS Version 1 Release 6, when DB2 becomes unavailable, you may also see this symptom in the LDAP server, until DB2 is restarted, at which time the LDAP server will reestablish connections with DB2.

SQL Native return code -99999 / SQL State 58004
can be caused by either a mixed DB2 environment (for example, running modules from different DB2 versions) or an invalid DSNAOINI file. For the DSNAOINI file, check that there are no sequence numbers and also that the file is FB80. Note that -99999/24000 may also indicate a mixed DB2 environment.

SQL State 58004 with RC=08 and Rsncode = 00f30011
SQL State 58004 with RC=08 and Rsncode = 00f30055
The problem could be that you need to increase your IDBACK ZPARM. For more information, refer to SQL INFOAPAR II12347. The following command reports on DB2 thread usage: *DB2subsys* DIS THREAD(*). Invoke this command periodically to determine peak thread usage to help in setting the IDBACK ZPARM value.

SQL Native return code -950 / SQL State 42705
can occur when the local database name indicated on the **servername** option in the LDAP configuration file or the data-source-name in the associated DATA SOURCE stanza of the CLI Initialization file (DSNAOINI) may be incorrect. The **servername** and data-source-name should contain the local database name, which is the name that was set during DB2 installation as *DB2 LOCATION NAME* on DB2 installation panel DSNTIPR for the DB2 subsystem. LOC1 is the default.

SQL Native return code -805 / SQL State 51002
When this error occurs, make sure that you have the appropriate hex character for the square brackets in the DSNAOINI file. These characters should be x'AD' and x'BD' for the left and right square brackets respectively. You may also need to rebind your DB2 plan, and validate that your **dbuserid** LDAP configuration option is valid.

DSNAOCLI file cannot be found
ensure you have installed the CLI component of DB2, which provides this file. Refer to "Getting DB2 installed and set up for CLI and ODBC" on page 13 for more information.

GLD0154I and GLD0155I
These errors are returned to LDAP from SQL calls, which return the specific error results from SQL, such as the native return code and state.

3. **APF/Program control problems**: If running LDAP with program control active, the LDAP server may generate GLD5002E error messages when a client attempts to execute LDAP utilities, such as ldapadd, ldapmodify, etc. The LDAP documentation only specifies that the datasets containing the LDAP DLLs (SYS1.SIEALNKE). The C Run Time Library (*.SCEERUN), and SYS1.LINKLIB need to be under program control. In addition, the following libraries should be added to this list:
   CSSLIB
   SCLBDLL
   SCSFMOD
   SCSFMOD1
   SDSNEXIT
   SDSNLOAD
   SEZALINK
4. **SSL setup problems**: See "Installing System SSL" on page 15 for SSL error codes.

gsk (SSL) error code 16
received on socket initialization may occur when you have a corrupted stash file, specified in the **sslKeyRingPWStashFile** LDAP configuration option.

GLD0069A Error –41 reported attempting SSL handshake.
-41 indicates 'GSK_SOC_BAD_V3_CIPHER'. Check the **sslCipherSpecs** LDAP configuration option. Ensure you have the support for the ciphers you indicated. For example, if the cipher value includes the mask for TRIPLE_DES_SHA_US, you would have to ensure the feature that supports TRIPLE_DES_SHA_US is installed.

GLD0069A Error –1 reported attempting SSL handshake.
-1 indicates 'GSK_ERR_NO_CIPHER. The client and server cannot agree on a supported cipherspec (encryption algorithm). This error can also occur if the client and server share no common cipher, or if no ciphers were defined.

Error '8' from gsk_initialize.
'8' indicates 'GSK_KEYFILE_NOT_FOUND'. The **listen** option for the LDAP server indicated that SSL support was requested, yet either no key file (kdb or keyring) was provided in the **sslKeyRingFile** LDAP configuration option, or the file specified could not be found. Check that your **sslKeyRingFile** option is specified and has a correct value.

Error '4' from gsk_initialize.
'4' indicates 'GSK_KEYFILE_BAD_PASSWORD'. This error commonly occurs when you are

using RACF a keyfile file and neglected to provide a NULL value for the **sslKeyRingFilePW** and **sslKeyRingFilePWStashFile** LDAP configuration options. Keyring support requires this to be NULL.

5. **Other**:

   GLD0144A The LDAP server encountered an error during configuration.
   LDAP debug tracing indicated iconv_open (UTF-8,IBM-1047) failed with errno=83 (ENOLOK) and returns LDAP return code x'52' from the xlate routine, which invoked the iconv routine. This problem can be corrected by making sure you have access to the SCEERUN C Runtime library in your LDAP server STEPLIB DD or from the LINKLIST.

   ocsf_encrypt: Encrypt data failed, pError=2120
   ocsf_test_MD5SHA1_encrypt: ocsf_encrypt failed, ocsf_rc=1
   If these messages are in the LDAP server log, the CDSA (ocsf) return code of 2120 indicates that ICSF is not active. See *z/OS Open Cryptographic Services Facility Service Provider Module Developer's Guide and Reference*, Appendix A, OCSF errors for more information. These messages are displayed to ensure that the ICSF code did not get lost or hidden somewhere, which is why they show up in the joblog.

Problems with a running LDAP server

How to debug

When a running z/OS LDAP server fails, you will receive an error message from LDAP or an Language Environment or system abend. An example of an error message from a running LDAP server would be:

```
GLD3112A An internal exception occurred creating schema cache, rc=90 Exception text is:
An attempt to allocate heap storage failed.
```

If Language Environment abends the LDAP server, a CEEDUMP will be produced in the LDAP server job spool. If a system abend is produced, typically an SVCDUMP is written and a message regarding the abend will be written to the system console. LDAP itself does not have any abend codes of its own, so never raises an abend condition. Program exceptions are caught by Language Environment's default ESPIE exits set up for all Language Environment enclaves, so Language Environment will typically generate a CEEDUMP for program exceptions as well.

Another symptom of a server failure would be a loop, which typically appears as high CPU utilization and waiting clients. A hang could also cause clients to wait or hang due to lack of server functionality. In the case of a suspected hang or loop, you would have to use the **DUMP** system command to dump the LDAP server address space.

For recreatable problems, to augment the information provided by any dumps produced, you can gather a server debug trace to provide to the Support Center along with the dumps. To obtain a debug trace of the LDAP server while it is running, you can enter a console **MODIFY** command to the LDAP server started task. For example:

```
f ldapsrv,appl=debug=2146959359
```

The LDAP server will write out a message indicating that the debug setting was put into effect. Once the problem is then recreated, you can turn off debug tracing by issuing. For example:

```
f ldapsrv,appl=debug=0
```

You must provide the following key information provided in the debug trace for entries with this reason code to the Support Center:

- the source file that produced the reason code
- the file's version number
- line of source in the source file that produced the reason code

For example, this may look like:

(tdbm_dll.c|1.77|778)

which indicates source file tdbm_dll.c at version 1.77 and source line 778 producing the reason code.

If the problem is not recreatable, or if it is not desirable to recreate the problem, then you forward the dump(s) obtained to the Support Center.

Common problems for a running LDAP server

There are a limited amount of instances of LDAP server hangs or failures. Typically, the server is setting errors or raising exceptions on behalf of failed client requests, but the server does not terminate. For example, when the LDAP server detects that TCP/IP has gone down, the server shuts itself down. One or both of the following messages may occur for every port on which the LDAP server was listening:

GLD0196A Attempt to accept() on IP address: *ip_address*, **port** *port_number* **failed; errno** *errno* **(***errno_string***).**
GLD0236I IP address: *ip_address*, **port** *port_number* **has been stopped.**

Debugging client problems from the LDAP server

How to debug

When the LDAP server receives incoming protocol, representing a request of work from a client, the server will return an LDAPResult construct, representing the results of the request. Depending on the design of the requesting client, the client may not interpret correctly, or react meaningfully. When any client's request success/error is in question, you can use the debug records written by the LDAP server, upon request, to indicate what return code and reason code was returned for a particular client's request. For recreatable problems, use the console **MODIFY** command to turn debug on, as mentioned above, and recreate the problem. You can then check the debug record output for errors. The easiest method for doing that is to search for string 'R00' which is the prefix for the reason codes, for example R001012 or R004026. The reason codes will provide the specific cause for the LDAP server returning the request in error. You can also proceed backwards in the record to find the incoming protocol/request as sent to the server. For successful requests, no reason code is returned, only a return code of '0'.

A client request can also fail from the client side itself, without the request even being sent to the LDAP server. In this case, the debugging of the client error will depend on the kind of client it is. When using the utilities (such as ldapmodify or ldapsearch) or 'C' APIs provided with the z/OS LDAP client, you can turn debug on to produce debug output. In the case of the utilities, you can use the debug option documented with the utility. For an application which uses the z/OS LDAP client 'C' APIs, you can set the debug level in the **LDAP_DEBUG** environment variable before starting the application. The client debug records are written to stderr, so ensure this output stream is captured. See the debug information in *z/OS Integrated Security Services LDAP Client Programming*.

Common problems for clients using the LDAP server

There are many return code and reason codes that can be returned from the LDAP server to the client in the case of an error. Below is a list of the most commonly reported problems, their documented reasons, and some recommendations for analysis:

R000128: Filter *filter* **is not supported**
Depending upon whether you are searching the SDBM, TDBM, or GDBM backends, the search filter indicated is in an invalid format. This error usually occurs on SDBM searches. There is a limited set of search filter formats for SDBM, which are documented in Chapter 16, "Accessing RACF information," on page 213.

R000137: *dn* **is not a valid RACF dn for bind.**
Refer to Chapter 16, "Accessing RACF information," on page 213 to describe the RACF-style *dn* format.

R001006: Value not found in schema. Value: *value*
This reason code typically occurs when defining a schema entry, and part of the new entry's

definition refers to another schema object *value* which is not yet defined. For example, you may try to add a new object class, definition that refers to an attribute that is not yet defined.

R001012: Attribute information not found for: *attr_type*
The reason code may occur when you use a DN that contains an attribute that is not yet defined in the schema. For example, you refer to DN `"xx=abc"` but attribute`'xx'` is not defined in the schema.

R001024: Abstract object class: *objectclass* **can not be leaf**
This reason code commonly occurs when you attempt to add an entry and the only **objectclass** provided for the entry is an abstract **objectclass**. Each entry in the database needs to be defined by at least one base structural **objectclass**. For more information regarding abstract and structural objectclasses, refer to Chapter 14, “LDAP directory schema,” on page 165.

R001028: More than one value specified for a single valued attribute type: *attr_type*
When defining values for attributes, you should notice whether an attribute is defined as single valued or multi-valued. Note that when an entry is added, the attributes and values in the RDN of the entry are always part of the entry. Thus, these attributes cannot have additional values in the entry unless the attributes are defined in the schema as multi-valued attributes. For example, if you are creating DN `"x=a,y=b,z=c"`, and attribute `'x'` in the RDN is a single valued attribute, you cannot further define attribute `'x'` to another value, you can only set it to the RDN value, which in this case is `'a'`.

R001038: Format of numeric object identifier not valid in value: *value*
As indicated, an OID value was provided that is invalid. The OID value must be numerals separated by a single period. For example: `1.26.3.1234`

R002001: Missing equal sign in DN component: *DN_component*
A valid DN is defined in Chapter 13, “Data model,” on page 161. Each component of a DN must have an equal sign between the attribute and value. For example, if a DN was specified as `"x=a,b,y=c"`, the second component, `'b'`, does not have an equal sign.

R002002: An escape sequence found in the following DN component is not valid: *DN_component*
Refer to “Distinguished name syntax” on page 162 to describe which characters are valid without escaping, as opposed to which characters need to be escaped with a \.

R004001: Unknown error occurred
This error will need to be reported to the Support Center.

R004026: Entry not found in the database
This is the most common reason code and simply states that the entry that you are searching for in the database does not exist. Also, returned with this reason code is the parent level found to exist, if any. For example, you have a leaf entry in the database as `z=c`, and you search for `x=a,y=b,z=c`. You receive a R004026 reason code returned on the search and also the indication that the closest ’parent’ entry that does exist is `z=c`.

R004028: The size limit for your search has been reached
The **sizelimit** LDAP configuration option will limit how many entries may be returned on a search request. This error indicates your search results have exceeded that limit. You may need to either refine your search base or filter to result in less entries being returned, or set the **sizelimit** configuration option to a larger value and restart the LDAP server.

Part 3. Messages

Chapter 31. LDAP server messages

This part contains the messages returned by the LDAP server. The messages are in alphanumeric order.

| The new LDAP server messages for z/OS Version 1 Release 6 are:
| • GLD0241A - GLD0266A
| • GLD3144A - GLD3153A
| • GLD7013A

LDAP server messages (0000)

GLD0002I Configuration file successfully read.

Severity: Information

Explanation: The LDAP server successfully read the configuration file.

System Action: The program continues.

Administrator Response: None.

GLD0004I Terminating slapd.

Severity: Information

Explanation: The LDAP server is ending, probably due to a SIGTERM signal.

System Action: The program ends.

Operator Response: None.

Administrator Response: None.

GLD0005E The ber_scanf failed during operation: *operation***.**

Severity: Eventual Action

Explanation: The LDAP server is unable to process the requested operation because of a failure when interpreting the request.

System Action: The program continues. The request fails.

Administrator Response: Correct the request and try again.

GLD0006E No values for type *type***.**

Severity: Eventual Action

Explanation: The LDAP server is unable to process the requested operation because no values were supplied for the specified type.

System Action: The program continues. The request fails.

Administrator Response: Correct the request and try again.

GLD0008E Unrecognized database type (*type***).**

Severity: Eventual Action

Explanation: The LDAP server encountered an error processing the configuration file. The specified database type is not supported.

System Action: The program continues. If all required parameters are not set correctly, the configuration will fail.

Administrator Response: Correct the configuration file and restart the server.

GLD0010I Reading configuration file *file_name***.**

Severity: Information

Explanation: The LDAP server is processing the specified configuration file.

System Action: The program continues.

Administrator Response: None.

GLD0011E *command_name***: line** *line_number***: option not valid or must appear inside a database definition. Ignored.**

Severity: Eventual Action

Explanation: The LDAP server encountered an error processing the configuration file. The option on the specified line is either not recognized or is associated with a database definition and must appear in the database section of the configuration file. The specified line is ignored. If the line does not belong in a database section, verify that the option is spelled correctly in the configuration file. Misspelled options may not be recognized by the server.

System Action: The program continues. If all required parameters are not set correctly, the configuration will fail and the server will not start. If the configuration parameter that failed is associated with a specific database backend, the server may start, but the backend may not be available. If a configuration parameter is ignored, the server may not operate as expected.

Administrator Response: Correct the configuration file and restart the server. Check the spelling of the configuration option on the specified line in the configuration file. Also verify that this option appears in the correct section of the configuration file.

GLD0012E *command_name*: **line** *line_number*: **incorrect configuration line. Ignored.**

Severity: Eventual Action

Explanation: The LDAP server encountered an error processing the configuration file because the specified line is not correct. The line is ignored.

System Action: The program continues. If all required parameters are not set correctly, the configuration will fail and the server will not start. If the configuration parameter that failed is associated with a specific database backend, the server may start, but the backend may not be available. If a configuration parameter is ignored, the server may not operate as expected.

Administrator Response: Correct the configuration file and restart the server.

GLD0013E *command_name*: **line** *line_number*: **incorrect number of parameters specified.**

Severity: Eventual Action

Explanation: The LDAP server encountered an error processing the configuration file because the specified line does not supply all of the required parameters.

System Action: The program continues. If all required parameters are not set correctly, the configuration will fail and the server will not start. If the configuration parameter that failed is associated with a specific database backend, the server may start, but the backend may not be available. If a configuration parameter is ignored, the server may not operate as expected.

Administrator Response: Correct the configuration file and restart the server.

GLD0014A Unable to open file *file_name*. **Try specifying the full path name.**

Severity: Immediate Action

Explanation: The LDAP server or a LDAP utility is unable to open the specified configuration file.

System Action: The program ends.

Operator Response: Correct the file name and restart the server, or contact the administrator.

Administrator Response: Determine the reason LDAP was unable to open the file. Possible reasons include misspelled file name, path to the file not fully specified, or file permissions incorrectly set. Correct the problem and restart the server.

GLD0015E *command_name*: **line** *line_number*: **unknown directive** *directive* **outside database definition. Ignored.**

Severity: Eventual Action

Explanation: The LDAP server encountered an error processing the configuration file because the specified line contains an unrecognized directive or keyword. The specified line is ignored.

System Action: The program continues. If all required parameters are not set correctly, the configuration will fail and the server will not start. If the configuration parameter that failed is associated with a specific database backend, the server may start, but the backend may not be available. If a configuration parameter is ignored, the server may not operate as expected.

Administrator Response: Correct the configuration file and restart the server.

GLD0016A A ber_alloc failed.

Severity: Immediate Action

Explanation: The LDAP server is unable to allocate the necessary storage to continue processing the request.

System Action: The program ends.

Operator Response: Increase the storage for the LDAP server and restart the server. If the problem persists, contact the service representative.

GLD0019A Error while trying to allocate memory.

Severity: Immediate Action

Explanation: The LDAP server is unable to allocate the necessary storage to continue processing.

System Action: The program ends.

Operator Response: Increase the storage for the LDAP server and restart the server. If the problem persists, contact the service representative.

GLD0020A Exceeded maximum number of connections, currently set at *max_connections*.

Severity: Immediate Action

Explanation: The LDAP server is unable to process the request because all available connections are in use.

System Action: The program continues. The request fails.

Administrator Response: Submit the request again.

GLD0021A Unable to create the necessary threads for *threadpool*.

Severity: Immediate Action

Explanation: The LDAP server is unable to obtain the necessary resources to create the threads in this threadpool. When this failure occurs during the startup of the LDAP Server, the program will end.

System Action: The program ends.

Operator Response: Verify the OMVS Kernel is operating correctly. Save the dump and contact the system programmer.

Administrator Response: If this error occurs with the commThreadPool, lower the number of `commThreads` in the configuration file or increase other tunable OMVS parameters such as MAXTHREADTASKS or MAXTHREADS. If this error occurs with the pcThreadPool, lower the number of `pcThreads` in the configuration file or increase other tunable OMVS parameters such as MAXTHREADTASKS or MAXTHREADS. If this error occurs with the replThreadPool, increase tunable OMVS parameters such as MAXTHREADTASKS or MAXTHREADS. If this problem persists, contact the service representative.

GLD0022I *LDAP_server_version* Starting slapd.

Severity: Information

Explanation: The LDAP server is starting.

System Action: The program continues.

Operator Response: None.

Administrator Response: None.

GLD0027A slapd unable to start because all backends failed to configure.

Severity: Immediate Action

Explanation: The LDAP server is unable to start because the defined backends have not configured successfully.

System Action: The program ends.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Check for other messages regarding errors during configuration. Correct the configuration file and restart the server.

GLD0028E Configuration error: server using port *port_number* for both SSL and non-SSL.

Severity: Eventual Action

Explanation: The LDAP server is listening on the specified port for both secure and nonsecure requests.

System Action: The program ends.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Correct the port values specified in the configuration file and restart the server.

GLD0038A A command that is not supported or is not valid was entered from the console.

Severity: Immediate Action

Explanation: The LDAP server received a command from the operator console that is not supported or is not correct.

System Action: The program continues.

Operator Response: Correct the command and try again.

GLD0039A The __console() function failed with errno *errno*.

Severity: Immediate Action

Explanation: The LDAP server received an unexpected return code from system function **__console()**. The *z/OS C/C++ Run-Time Library Reference*, SC28-1663 contains more information about the **errno**. Error messages normally written to the console will only appear in the log.

System Action: The program continues.

Operator Response: Save the dump, if any, and contact the system programmer. If the problem persists, contact the service representative.

GLD0041A An administrator DN must be specified in the adminDn line.

Severity: Immediate Action

Explanation: The LDAP server encountered a blank `adminDN` parameter in the configuration file.

System Action: The program ends.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Correct the `adminDN` parameter in the configuration file and restart the server.

GLD0043A Unable to connect to replica *replica_name* on port *port_number*. Please verify that the replica is started.

Severity: Immediate Action

Explanation: The LDAP server was unable to connect to the specified server to perform replication.

System Action: The program continues. Replication to the specified server cannot continue.

Operator Response: Verify that the replica server is started.

Administrator Response: Verify that the replica server is started, or contact the operator to start the replica server. Verify that the replica server information is correct.

GLD0044E Error *error_value* occurred during replication to *replica_name*: add failed on entry *distinguished_name*.

Severity: Eventual Action

Explanation: The LDAP server was unable to replicate the specified add operation to the replica server. The replication attempt will be tried again.

System Action: The program continues.

Administrator Response: Verify that the replica server information is correct. If the failure continues, contact the service representative.

GLD0045E Error *error_value* occurred during replication to *replica_name*: delete failed on entry *distinguished_name*.

Severity: Eventual Action

Explanation: The LDAP server was unable to replicate the specified delete operation to the replica server. The replication attempt will be tried again.

System Action: The program continues.

Administrator Response: Verify that the replica server information is correct. If the failure continues, contact the service representative.

GLD0046E Error *error_value* occurred during replication to *replica_name*: modrdn failed on entry *distinguished_name*.

Severity: Eventual Action

Explanation: The LDAP server was unable to replicate the specified modify RDN operation to the replica server. The replication attempt will be tried again.

System Action: The program continues.

Administrator Response: Verify that the replica server information is correct. If the failure continues, contact the service representative.

GLD0047E Error *error_value* occurred during replication to *replica_name*: modify failed on entry *distinguished_name*.

Severity: Eventual Action

Explanation: The LDAP server was unable to replicate the specified modify operation to the replica server. The replication attempt will be tried again.

System Action: The program continues.

Administrator Response: Verify that the replica server information is correct. If the failure continues, contact the service representative.

GLD0048A Creation of socket failed; errno *errno* (*errno_string*).

Severity: Immediate Action

Explanation: The LDAP server received the specified error from system function **socket()**. Refer to the *z/OS C/C++ Run-Time Library Reference*, SC28-1663 for an explanation of the **errno** returned.

System Action: Initialization continues for all other requested communication interfaces. If no communication interface is successfully initialized, the program ends.

Operator Response: Ensure TCP/IP is operating correctly. Save the diagnostic information and contact the system programmer.

Administrator Response: If the problem persists, contact the service representative.

GLD0049A Attempt to setsockopt(*socket_option*) failed; errno *errno* (*errno_string*).

Severity: Immediate Action

Explanation: The LDAP server received the specified error from system function **setsockopt()**. Refer to the *z/OS C/C++ Run-Time Library Reference*, SC28-1663 for an explanation of the **errno** returned.

System Action: The program continues.

Operator Response: Ensure TCP/IP is operating correctly. Save the diagnostic information and contact the system programmer.

Administrator Response: If the problem persists, contact the service representative.

GLD0050A Attempt to bind failed; errno *errno* (*errno_string*).

Severity: Immediate Action

Explanation: The LDAP server received an error from system function **bind()**. Refer to the *z/OS C/C++*

Run-Time Library Reference, SC28-1663 for an explanation of the **errno** returned.

System Action: Initialization continues for all other requested communication interfaces. If no communication interface is successfully initialized, the program ends.

Operator Response: Ensure TCP/IP is operating correctly. Save the diagnostic information and contact the system programmer.

Administrator Response: If the problem persists, contact the service representative.

GLD0051A The listen() failed; errno *errno* **(***errno_string***).**

Severity: Immediate Action

Explanation: The LDAP server received an error from system function **listen()**. Refer to the *z/OS C/C++ Run-Time Library Reference*, SC28-1663 for an explanation of the **errno** returned.

System Action: Initialization continues for all other requested communication interfaces. If no communication interface is successfully initialized, the program ends.

Operator Response: Ensure TCP/IP is operating correctly. Save the diagnostic information and contact the system programmer.

Administrator Response: If the problem persists, contact the service representative.

GLD0052I Configuration read securePort *port_number***.**

Severity: Information

Explanation: The LDAP server has assigned the `securePort` to the specified value based on the value read from the configuration file.

System Action: The program continues.

Administrator Response: None.

GLD0053I Configuration read security of *security_value***.**

Severity: Information

Explanation: The LDAP server has assigned the `security` to the specified value based on the value read from the configuration file.

System Action: The program continues.

Administrator Response: None.

GLD0060I Non-SSL port override from command line, value = *value***.**

Severity: Information

Explanation: The LDAP server has assigned the nonsecure port to the specified value based on the value supplied on the command line.

System Action: The program continues.

Administrator Response: None.

GLD0061I SSL port override from command line, value = *value***.**

Severity: Information

Explanation: The LDAP server has assigned the secure port to the specified value based on the value supplied on the command line.

System Action: The program continues.

Administrator Response: None.

GLD0063A Error *error_code* **reported by SSL initialization.**

Severity: Immediate Action

Explanation: The LDAP server was unable to complete initialization required for secure communications.

System Action: Communication interfaces that require SSL will not be activated. Communication interfaces that support both secure and non-secure communication will be activated for non-secure communication only. If no interfaces are activated then the program will end.

Operator Response: The error code is documeted in the **ldapssl.h** file shown in *z/OS Integrated Security Services LDAP Client Programming*. If the error description indicates a correctable error then correct it and restart the server. Otherwise, contact the service representative.

GLD0064A SSL detected an error reading *file_name***.**

Severity: Immediate Action

Explanation: The LDAP server was unable to read the key database file or key ring required for secure communications. Secure communications cannot continue.

System Action: Communication interfaces that require SSL will not be activated. Communication interfaces that support both secure and non-secure communication will be activated for non-secure communication only. If no interfaces are activated then the program will end.

Operator Response: Verify that the file system is operating correctly.

Administrator Response: Ensure that the file permissions on the SSL key ring file are correct.

GLD0065A SSL encountered an error opening *file_name*.

Severity: Immediate Action

Explanation: The LDAP server was unable to open the key database file or key ring required for secure communications. Secure communications cannot continue.

System Action: Communication interfaces that require SSL will not be activated. Communication interfaces that support both secure and non-secure communication will be activated for non-secure communication only. If no interfaces are activated then the program will end.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Ensure that the file permissions on the SSL key database file are correct and that it exists.

GLD0066A SSL key database file *file_name* is in an unknown format.

Severity: Immediate Action

Explanation: The System SSL runtime library is unable to decrypt a key database entry. Either the supplied database password is incorrect or the database is damaged. Secure communications cannot continue.

System Action: Communication interfaces that require SSL will not be activated. Communication interfaces that support both secure and non-secure communication will be activated for non-secure communication only. If no interfaces are activated then the program will end.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Ensure that the correct key database password is specified in the configuration file. Recreate the database if the error persists.

GLD0067A The password supplied for SSL key database file *file_name* is not correct.

Severity: Immediate Action

Explanation: The LDAP server found that the password supplied for the key database file is incorrect. Secure communications cannot continue.

System Action: Communication interfaces that require SSL will not be activated. Communication interfaces that support both secure and non-secure communication will be activated for non-secure communication only. If no interfaces are activated then the program will end.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Correct the password for the key database file in the configuration file or the password stash file and restart the server.

GLD0069A Error *error_code* reported attempting SSL handshake.

Severity: Immediate Action

Explanation: The LDAP server was unable to complete initialization of the socket required for secure communications.

System Action: The program continues.

Operator Response: The error code is documented in the **ldapssl.h** file shown in *z/OS Integrated Security Services LDAP Client Programming*. If the error description indicates a client error then correct it and restart the client. If the error description indicates a correctable server error then correct it and restart the server. Otherwise, contact the service representative.

GLD0070A An incorrect SSL certificate label *label* was specified.

Severity: Immediate Action

Explanation: The LDAP server found that the certificate label supplied for the key database file or key ring is incorrect. Secure communications cannot continue.

System Action: The program continues. Any communication inferface related to SSL will not continue.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Correct the certificate label in the configuration file and restart the server.

GLD0072A No SSL ciphers matched both the client and server cipher specifications.

Severity: Immediate Action

Explanation: The client and server cipher specifications do not contain at least one value in common. This error can also occur if no SSL protocols are enabled or if all of the enabled protocols have empty cipher specifications.

System Action: The program continues. Secure communication between the server and the client will fail.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Ensure that the client and the server have at least one cipher specification in common. If the server needs additional ciphers then

update the cipher specification in the configuration file and restart the server. If the client needs additional ciphers then correct that condition and retry the client.

GLD0073A The default SSL certificate has expired in *file_name*.

Severity: Immediate Action

Explanation: The LDAP server found that the default certificate in the key database file or key ring is no longer valid.

System Action: The program continues. Any communication inferface related to SSL will not continue.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Refresh the certificate and restart the server.

GLD0075A The default SSL certificate is of an unsupported type in *file_name*.

Severity: Immediate Action

Explanation: The LDAP server found that the default certificate in the key database file or key ring is not the correct type.

System Action: The program continues. Any communication inferface related to SSL will not continue.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Obtain a supported certificate and restart the server.

GLD0076A No SSL certificate exists in *file_name*.

Severity: Immediate Action

Explanation: The LDAP server found that no certificate exists in the key database file or key ring.

System Action: The program continues. Any communication inferface related to SSL will not continue.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Refresh the certificate and restart the server.

GLD0077A The underlying socket was closed.

Severity: Immediate Action

Explanation: The LDAP server was unable to complete secure communications because the socket is closed.

System Action: The program continues.

Operator Response: None.

Administrator Response: Contact the service representative.

GLD0083I Successfully reconnected to replica host *replica_name* on port *port_number*.

Severity: Information

Explanation: The LDAP server was able to restart replication to the specified replica server.

System Action: The program continues.

Administrator Response: None.

GLD0084A Connection to replica *replica_name* on port *port_number* has failed. Verify that the replica is started.

Severity: Immediate Action

Explanation: The LDAP server detected that the connection to the specified replication server ended.

System Action: The program continues. Replication to the specified server cannot continue.

Operator Response: Verify that the replica server is started.

Administrator Response: Verify that the replica server is started or contact the operator to start the replica server. Verify that the replica server information is correct.

GLD0089E Attention: configuration file *file_name* is empty.

Severity: Eventual Action

Explanation: The LDAP server encountered an empty configuration file.

System Action: The program ends if the empty file is the configuration file. The program continues if the empty file is an included configuration file.

Administrator Response: Ensure the correct configuration files are specified.

GLD0090A A modify command that failed was entered from the console: *command*.

Severity: Immediate Action

Explanation: The LDAP server received a modify command from the operator, but the command failed. Either the syntax was wrong or the processing invoked by the command experienced a problem.

System Action: The program continues.

Operator Response: Correct the syntax or determine why the processing of the command failed.

GLD0091I Successfully set debug level to *debug_level* from console command.

Severity: Information

Explanation: The LDAP server set the debug level to the specified value.

System Action: The program continues.

Operator Response: None.

Administrator Response: None.

GLD0092A Unable to open any configuration file. No configuration file specified at startup, tried DDname *DD_name* and default name *file_name*.

Severity: Immediate Action

Explanation: The LDAP program is unable to open any configuration file. No configuration file name was specified at startup using the **-f** option. The program tried the specified DDname and then the default configuration file, but was unable to find any configuration file. The program cannot start without a configuration file.

System Action: The program ends.

Operator Response: Correct the configuration file name and restart the server or contact the administrator.

Administrator Response: Correct the file name or permissions and restart the server.

GLD0107A The configuration file produces a loop within the include statements.

Severity: Immediate Action

Explanation: A loop was detected during the processing of the included configuration files during server startup.

System Action: The program ends.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Check the configuration file include statements.

GLD0108E No object class was specified for entry *entry_dn*.

Severity: Eventual Action

Explanation: All entries must specify an object class.

System Action: The program continues. The request fails.

Administrator Response: Verify the LDIF file syntax.

GLD0109E The required attribute *attribute_name* is missing for entry *entry_dn*.

Severity: Eventual Action

Explanation: An attribute required by an object class specified in this entry was not provided.

System Action: The program continues. The request fails.

Administrator Response: Verify the LDIF file syntax. Refer to the schema files for a list of the attributes required for each object class.

GLD0110E The attribute *attribute_name* is not allowed for entry *entry_dn*.

Severity: Eventual Action

Explanation: The specified attribute is not allowed by the object classes specified in this entry.

System Action: The program continues. The request fails.

Administrator Response: Verify the LDIF file syntax. Refer to the schema files for a list of the attributes allowed for each object class.

GLD0111E An error occurred while processing schema file *file_name*: line *line_number* syntax :*oc clause* ::= objectclass *ocname* [requires *attrlist*] [allows *attrlist*]

Severity: Eventual Action

Explanation: The LDAP server object class schema file contains a schema definition that is not valid.

System Action: The program ends.

Administrator Response: Correct the object class definition in the specified schema file and restart the server.

GLD0114I A client sent nonsecured communications to the SSL port.

Severity: Information

Explanation: The LDAP server determined that a client sent unencrypted data to the secure port. The request from the client is ended.

System Action: The program continues. The request fails.

Administrator Response: If a command utility such as **ldapsearch** is called by the client and secure communications is intended, make sure the **-Z** (use secure communications) parameter is specified. If the client does not intend to use secure communications, then specify the nonsecure port. If secure communications is intended, make sure the client calls

ldap_ssl_start. If secure communications is not intended then specify the nonsecure port.

GLD0115I **Workload Manager enablement initialization successful for group=***sysplex_groupname***, server=***sysplex_servername* **on host** *host_name***.**

Severity: Information

Explanation: The LDAP server successfully registered with Sysplex Workload Manager. The LDAP server will operate in multi-server mode. Multiple concurrent servers may be running on a given z/OS image, and multiple concurrent servers may be running on multiple z/OS images in a parallel sysplex, all of which use the same LDAP DB2 database. Note that no replication may be performed by any of these servers. If replication is desired, only one server instance may be running which uses a given LDAP DB2 database, and the server must be configured to run in single-server mode.

System Action: The program continues.

Operator Response: None.

Administrator Response: None.

GLD0116E **Workload Manager enablement initialization failed for group=***sysplex_groupname***, server=***sysplex_servername* **on host** *host_name***. No Workload Manager support will be activated for this server. RC =** *return_code***.**

Severity: Eventual Action

Explanation: The LDAP server registration with Sysplex Workload Manager failed.

System Action: The program continues.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Contact the service representative and provide the return code in this message.

GLD0117I **Workload Manager enablement termination successful for group=***sysplex_groupname***, server=***sysplex_servername* **on host** *host_name***.**

Severity: Information

Explanation: The LDAP server successfully unregistered from Sysplex Workload Manager.

System Action: The program continues.

Administrator Response: None.

GLD0118E **Workload Manager enablement termination failed for group=***sysplex_groupname***, server=***sysplex_servername* **on host** *host_name***. RC =** *return_code***.**

Severity: Eventual Action

Explanation: The LDAP server was unable to unregister from Sysplex Workload Manager.

System Action: The program continues.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: If the problem persists, contact the service representative.

GLD0119E **Workload Manager enablement initialization failed for group=***sysplex_groupname***, server=***sysplex_servername***. No Workload Manager support will be activated for this server. RC =** *return_code***.**

Severity: Eventual Action

Explanation: The LDAP server was unable to register with Sysplex Workload Manager. This is an internal program error. Contact the service representative with the RC value displayed.

System Action: The program continues.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Contact the service representative.

GLD0120E **Workload Manager enablement termination failed for group=***sysplex_groupname***, server=***sysplex_servername***. RC =** *return_code***.**

Severity: Eventual Action

Explanation: The LDAP server was unable to deregister from Sysplex Workload Manager. This is an internal program error. Contact the service representative with the RC value displayed.

System Action: The program continues.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Contact the service representative.

GLD0121E The object class *object_class* specified for entry *distinguished_name* is not defined in the schema.

Severity: Eventual Action

Explanation: An object class specified in a request for the given DN is not defined in the current schema.

System Action: The program continues. The request fails.

Administrator Response: Verify the object class in the request and the object classes defined in the schema configuration files and try the request again.

GLD0122I Slapd is ready for requests.

Severity: Information

Explanation: The LDAP server is listening and is ready for requests.

System Action: The program continues.

Operator Response: None.

Administrator Response: None.

GLD0123E Workload Manager enablement initialization failed for group=*sysplex_groupname*, server=*sysplex_servername* on host *host_name* because this group/server combination is already registered on this host. No Workload Manager support will be activated for this server. RC = *return_code*.

Severity: Eventual Action

Explanation: The LDAP server registration with Sysplex Workload Manager failed because another server has already been registered by the same name in the same sysplex group.

System Action: The program continues.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Ensure that the `sysplexServerName` in the server configuration file (slapd.conf) is unique among all servers started with the same `sysplexGroupName`. If the problem persists, contact the service representative.

GLD0124I Dynamic workload management enabled. Server will operate in multi-server mode. sysplexServerName = *sysplex_server_name*, sysplexGroupName = *sysplex_group_name*.

Severity: Information

Explanation: The LDAP server will operate in multi-server mode. Multiple concurrent servers may be running on a given z/OS image, and multiple concurrent servers may be running on multiple z/OS images in a parallel sysplex, all of which use the same LDAP DB2 database. Note that no replication may be performed by any of these servers. If replication is desired, only one server instance may be running which uses a given LDAP DB2 database, and the server must be configured to run in single-server mode.

System Action: The program continues.

Administrator Response: None.

GLD0125E Only one of the sysplexGroupName or sysplexServerName keywords is present in the configuration file.

Severity: Eventual Action

Explanation: One of the sysplex keywords `sysplexGroupName` or `sysplexServerName` was found in the server configuration file. If either of these keywords is present, the other must also be present, and both keywords must be accompanied by non-null arguments.

System Action: The program ends.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Correct the configuration file and restart the server.

GLD0126A The length of *sysplex_keyword* must be less than or equal to *maximum_argument_length*.

Severity: Immediate Action

Explanation: The length of the `sysplexGroupName` argument must not be greater than 18 characters. The length of the `sysplexServerName` argument must not be greater than 8 characters.

System Action: The program ends.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Correct the configuration file and restart the server.

GLD0127A Workload Manager enablement initialization failed due to memory allocation error. No Workload Manager support will be activated for this server.

Severity: Immediate Action

Explanation: The LDAP server could not allocate memory needed to register with Workload Manager. The LDAP server will continue to operate, but no Workload Manager functions will be available.

System Action: The program continues.

Operator Response: Increase the storage for the LDAP server and restart the server. If the problem persists, contact the service representative.

GLD0128A The SSL certificate sent by the client or the server certificate in *file_name* is not valid.

Severity: Immediate Action

Explanation: The LDAP server found that the certificate sent by the client or the certificate in the key database file or key ring is not valid.

System Action: The program continues. The request fails.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Verify that the server certificate in the key database file or key ring is still valid. Things to look for in the server certificate information are the issuer Certificate Authority and the expiration time of the certificate.

GLD0129E The value for the 'maxConnections' option is out of range (*min_Connection*, *max_Connection*). The default (*max_Connection*) will be used.

Severity: Eventual Action

Explanation: The LDAP server is unable to process the request because the value of `maxConnections` is out of range. The server will continue with the default value

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0131E The value for the sizelimit option is not numeric. The default (*default_sizelimit*) will be used.

Severity: Eventual Action

Explanation: The LDAP server determined that the value specified for the `sizelimit` option of the configuration file is not numeric. The server will continue with the default value.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0132E The value for the timelimit option is not numeric. The default (*default_timelimit*) will be used.

Severity: Eventual Action

Explanation: The LDAP server determined that the value specified for the `timelimit` option of the configuration file is not numeric. The server will continue with the default value.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0135E The value specified for the readonly option is not valid. The default (*default_readonly*) will be used.

Severity: Eventual Action

Explanation: The LDAP server encountered an error during processing of the `readonly` option of the configuration file. The default value will be used.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0136A The value specified for the masterserverDN option is not valid.

Severity: Immediate Action

Explanation: The LDAP server encountered an error during processing of the `masterserverDN` option of the configuration file. The `masterserverDN` cannot have a NULL value.

System Action: The program continues. However, configuration may fail. Replication will not be configured.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Correct the configuration file and restart the server.

GLD0137A The value specified for the masterserverDN option is 'cn=Anybody'. This is not permitted.

Severity: Immediate Action

Explanation: The LDAP server encountered an error during processing of the `masterserverDN` option of the

configuration file. The masterserverDN cannot be 'cn=Anybody'.

System Action: The program ends.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Correct the configuration file and restart the server.

GLD0138E The value for the maxConnections option is not numeric. The default (*default_max_connections*) will be used.

Severity: Eventual Action

Explanation: The LDAP server determined that the value specified for the maxConnections option of the configuration file is not numeric. The server will continue with the default value.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0141E A backend (*backend_address*) of type *backend_type* failed to configure.

Severity: Eventual Action

Explanation: The LDAP server encountered an error during configuration of the specified backend. This backend will not be available when the server starts.

System Action: The program continues. Other backends that configure successfully will be available. If no backends configure successfully, an additional message will appear and the program will end.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Check for other messages regarding errors during configuration. Correct the configuration file and restart the server.

GLD0142E The value *URL_value* for the *configuration_option* is not a valid URL or cannot be resolved.

Severity: Eventual Action

Explanation: The LDAP server determined that the value provided for the specified option of the configuration file is not in a valid LDAP URL format. Another possible problem is that the hostname or IP address specified in the value can not be properly resolved because a Domain Name Server or TCP/IP is not available.

System Action: The program continues. However, configuration may fail.

Operator Response: None.

Administrator Response: Correct the configuration file parameter and restart the server.

GLD0143A The LDAP program cannot create required configuration structures.

Severity: Immediate Action

Explanation: The LDAP program encountered an error when establishing a required configuration structure. The program cannot start.

System Action: The program ends.

Operator Response: Increase the storage for the LDAP server and restart the server. If the problem persists, contact the service representative.

Administrator Response: Ensure adequate memory for the program. Correct any other reported errors and restart the program.

GLD0144A The LDAP server encountered an error during configuration.

Severity: Immediate Action

Explanation: The LDAP server encountered an error during configuration. The program cannot start. See other messages regarding errors.

System Action: The program ends.

Operator Response: Contact the LDAP Administrator or see the Administrator Response. If the problem persists, contact the service representative.

Administrator Response: Examine other messages and correct any other reported errors. Restart the LDAP server.

GLD0146E The value for the -s command line option is not numeric. Value is ignored.

Severity: Eventual Action

Explanation: The LDAP server determined that the value specified for the **-s** option on the command line invocation of the program is not numeric. The value is ignored. The **-s** option identifies the secure port.

System Action: The program continues.

Administrator Response: Correct the command line option and restart the server.

GLD0147E The value for the -p command line option is not numeric. Value is ignored.

Severity: Eventual Action

Explanation: The LDAP server determined that the value specified for the **-p** option on the command line invocation of the program is not numeric. The value is ignored. The **-p** option identifies the port.

System Action: The program continues.

Administrator Response: Correct the command line option and restart the server.

GLD0148A The dllload function failed loading path *loadpath* with errno=*errno*, error string=*error_string*.

Severity: Immediate Action

Explanation: The LDAP server was unable to load the requested library. Refer to the *z/OS C/C++ Run-Time Library Reference*, SC28-1663 for an explanation of the **errno**.

System Action: The program continues. However, configuration may fail.

Operator Response: Contact the LDAP Administrator or see the Administrator Response. If the problem persists, contact the service representative.

Administrator Response: Correct the error and restart the server.

GLD0149E The value specified for the extendedGroupSearching option is not valid. The default value of (*default_extendedGroupSearching*) will be used.

Severity: Eventual Action

Explanation: The LDAP server encountered an error during processing of the `extendedGroupSearching` option of the configuration file. The default value will be used.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0150A The LDAP program requires a TDBM backend but none is configured.

Severity: Immediate Action

Explanation: A TDBM backend must be configured for this LDAP program to run. Either no TDBM backend was configured or an error was encountered during TDBM configuration. The LDAP program cannot start.

System Action: The program ends.

Operator Response: None.

Administrator Response: Ensure TDBM configuration is present. Correct any other reported errors and restart the program.

GLD0151A Object class *object_class* requires attribute type *attribute_type* which is not defined.

Severity: Immediate Action

Explanation: The LDAP server encountered an error during processing of the object class definitions.

System Action: The program ends.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Correct the schema definition by correcting or removing the attribute type from the object class definition or adding the attribute type to the schema and restart the server.

GLD0154E Error code *error_code* from odbc string: *"odbc_string" substring* .

Severity: Eventual Action

Explanation: An error occurred when performing DB2 operations. The request fails. There may be an additional message with information about SQL return codes.

System Action: The program continues. The request fails.

Administrator Response: Evaluate and resolve the DB2 problem with the information provided. If the problem cannot be resolved, contact the service representative.

GLD0155E ODBC error, SQL data is: native return code=*SQL_code*, SQL state=*SQL_state*, SQL message=*SQL_message*.

Severity: Eventual Action

Explanation: An error occurred when performing DB2 operations. The information in the message is the data available from SQL at the time of the error.

System Action: The program continues. The request fails.

Administrator Response: Evaluate and resolve the DB2 problem with the information provided. If the problem cannot be resolved, contact the service representative.

GLD0156A Unable to normalize administrator DN: *admin_dn*.

Severity: Immediate Action

Explanation: The LDAP program encountered an error while attempting to normalize the administrator DN from the configuration file. A possible reason for the error is no equal sign in the relative DN. The *admin_dn* will be **binary** if unable to convert the string.

System Action: The program continues. However, configuration may fail.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Determine the reason for

the error. Correct the `adminDN` in the configuration file and restart the server.

GLD0157A Unable to normalize master server DN: *master_server_dn***.**

Severity: Immediate Action

Explanation: The LDAP program encountered an error while attempting to normalize the master server DN from the configuration file. A possible reason for the error is no equal sign in the relative DN. The *admin_dn* will be **binary** if unable to convert the string.

System Action: The program continues. However, configuration may fail.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Determine the reason for the error. Correct the `masterServerDN` in the configuration file and restart the server.

GLD0158A Unable to normalize suffix: *suffix***.**

Severity: Immediate Action

Explanation: The LDAP program encountered an error while attempting to normalize a suffix from the configuration file. A possible reason for the error is no equal sign in the relative DN. The *admin_dn* will be **binary** if unable to convert the string.

System Action: The program continues. However, configuration may fail.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Determine the reason for the error. Correct the `suffix` in the configuration file and restart the server.

GLD0159A Unknown exception caught during configuration.

Severity: Immediate Action

Explanation: The LDAP program encountered an unknown error during configuration.

System Action: The program ends.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: It may be possible to determine the error using LDAP debug tracing. If unable to resolve the problem using the debug trace, the service representative should be contacted.

GLD0160E A replica with DN: *replica_dn***, host name:** *hostname***, and port:** *port* **duplicates an existing replica with DN:** *replica_dn***. The existing replica will be used.**

Severity: Eventual Action

Explanation: The identified replica object has the same host name and port as an existing replica object, and the bind DN and credentials also match those in the existing replica object. The existing replica object will be used to control replication. While the new replica will be added to the directory, it will not be used to control replication.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0161E A replica with DN: *replica_dn***, host name:** *hostname***, and port:** *port* **attempted to duplicate an existing replica with DN:** *replica_dn***. The new replica is not added to the directory.**

Severity: Eventual Action

Explanation: The identified replica object has the same host name and port as an existing replica object. One of the following reasons applies:

- The new replica object has the same host and port as another replica object in the directory, but the bind DN and credentials do not match those in the existing replica object.
- The new replica object duplicates another replica object that was recently deleted from the directory. Clean up processing for the recently removed replica object has not completed.

The existing replica object remains active.

System Action: The program continues. However, the duplicate replica definition is not added to the directory.

Administrator Response: A replica with the same host name and port as an existing replica can only be added if the bind DN and credentials in the two entries also match. Correct the replicaObject entry being added and try the add again. If a replica object has been recently deleted, wait ten minutes to allow clean up processing to complete and try the request again.

GLD0162I Backend of type: *backend_type* **serving suffix:** *suffix_dn* **does not support reporting of backend capabilities.**

Severity: Information

Explanation: The LDAP server has loaded a backend which does not contain the necessary programming support to report the capabilities of that backend.

System Action: The program continues.

Administrator Response: None.

GLD0163I Backend capability listing follows:

Severity: Information

Explanation: The LDAP server has completed loading backends and will report the capabilities of those backends which have implemented the necessary programming support to do so.

System Action: The program continues.

Administrator Response: None.

GLD0164I Backend capability listing ended.

Severity: Information

Explanation: The LDAP server has completed reporting the capabilities of those backends which have implemented the necessary programming support to do so.

System Action: The program continues.

Administrator Response: None.

GLD0165I Capability: *attribute_type* **Value:** *attribute_value*

Severity: Information

Explanation: The LDAP server is reporting the capabilities of backends which have implemented the necessary programming support to do such reporting. The Capability indicates what capability attribute is being reported for the backend, and the Value expresses the attribute value for that particular capability. Following is a partial list of common capabilities which may be reported, accompanied by a brief description of each:

- Backend ID - a string used to uniquely identify the backend.
- Build Date and Time - the date on which the backend was compiled, in the format yyyy-mm-dd-hh.mm.ss.000000.
- Apar Level - if the backend was compiled as part of an APAR, the APAR number is reported by the backend.
- Release - a release identifier associated with the backend.
- Version - a version identifier associated with the backend.
- Dialect - an indicator of the internal interface version to which the backend is programmed.
- Berdecoding - indicates whether the server front-end code should decode the incoming BER value in string format or in binary format before passing the values to the backend.
- ExtendedGroupSearching - indicates whether the backend is capable of participation in extended group membership searching on a client bind request.
- SupportedControls - lists which controls are supported by the backend.
- SupportedExtension - lists which extendedRequests and extendedResponses are supported by the backend.

System Action: The program continues.

Administrator Response: None.

GLD0166I Backend type: *backend_type***, Backend ID:** *backend_identifier*

Severity: Information

Explanation: The LDAP server is reporting the capabilities of backends. Backend type is the type of backend configured in the server configuration file (that is, sdbm, tdbm, gdbm, xpdir, and so on). Backend ID is the identity of the backend.

System Action: The program continues.

Administrator Response: None.

GLD0167I End of capability listing for Backend type: *backend_type***, Backend ID:** *backend_identifier***.**

Severity: Information

Explanation: The LDAP server has completed reporting the capabilities of the current backend. Backend type is the type of backend configured in the server configuration file (that is, sdbm, tdbm, gdbm, xpdir, and so on). Backend ID is the identity of the backend.

System Action: The program continues.

Administrator Response: None.

GLD0168I Backend type *backend_type***: Backend suffix:** *backend_suffix* **lacking required capability attribute(s) or value(s) and will be unloaded.**

Severity: Information

Explanation: The LDAP server detected a backend which implemented the necessary programming support to report capabilities of the backend, and which is lacking one or more required capability attributes or attribute values. This backend will be unloaded and will be unavailable to the server.

System Action: The program continues.

Administrator Response: Under normal operation, this error should not occur. Contact the service representative.

GLD0169A Acquisition or check of backend capabilities failed for backend of type: *backend_type*.

Severity: Immediate Action

Explanation: During its check of backend capabilities, the LDAP server encountered an unrecoverable error.

System Action: The program continues. Backends that configure successfully will be available. If no backends configure successfully, an additional message will appear and the program will end.

Administrator Response: If the problem persists for the same `Backend Type` remove the `database` stanza for this `Backend Type` from the server configuration file to permit starting the server without this backend (if desired), and contact the service representative.

GLD0170I Kerberos authentication support has been enabled.

Severity: Information

Explanation: The LDAP server will be configured for Kerberos authentication.

System Action: The program continues.

Administrator Response: None.

GLD0171I Kerberos authentication support has not been enabled.

Severity: Information

Explanation: The LDAP server will not be configured for Kerberos authentication.

System Action: The program continues.

Administrator Response: None.

GLD0173E Server was unable to acquire Kerberos credentials.

Severity: Eventual Action

Explanation: Kerberos credentials could not be obtained for the server. Kerberos support will be disabled.

System Action: The program continues.

Operator Response: Make sure the Kerberos KDC has been started.

Administrator Response: If not using Kerberos, remove Kerberos information from the configuration file and restart the server.

GLD0174E Server is unable to acquire Kerberos credentials without a principal.

Severity: Eventual Action

Explanation: The server's Kerberos principal name `serverKrbPrinc` was not specified in the configuration file. Kerberos support will be disabled.

System Action: The program continues.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Add the server's Kerberos principal to the configuration file and restart the server.

GLD0175E The value for the 'commThreads' option is not numeric. The default (*default_comm_threads*) will be used.

Severity: Eventual Action

Explanation: The LDAP server determined that the value specified for the `commThreads` option of the configuration file is not numeric. The server will continue with the default value.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0176E The value for the 'commThreads' option is out of range (*min_threads*, *max_threads*). The default (*default_comm_threads*) will be used.

Severity: Eventual Action

Explanation: The LDAP server encountered an error when processing the `commThreads` parameter of the configuration file. The server will continue with the default value.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0177E The value for the 'idleConnectionTimeout' option is not numeric. The default of indefinite (0) will be used.

Severity: Eventual Action

Explanation: The LDAP server determined that the value specified for the `idleConnectionTimeout` option of the configuration file is not numeric. The server will continue with the default value.

System Action: The program continues. Connections will not timeout.

Administrator Response: Correct the configuration file and restart the server.

GLD0178E The value for the 'idleConnectionTimeout' option is less than *min_timeout*. The default of indefinite (0) will be used.

Severity: Eventual Action

Explanation: The LDAP server encountered an error when processing the `idleConnectionTimeout` parameter of the configuration file. The server will continue with the default value.

System Action: The program continues. Connections will not timeout.

Administrator Response: Correct the configuration file and restart the server.

GLD0179A An internal error was detected in the communication area start-up table for tower '*tower_ID*'.

Severity: Immediate Action

Explanation: The LDAP server detected an internal error in the communication area start-up table.

System Action: The program continues.

Operator Response: Save the diagnostic information and contact the administrator.

Administrator Response: Contact the service representative.

GLD0181E The '*threadParm*' parameter is no longer supported.

Severity: Eventual Action

Explanation: This thread parameter is now obsolete and is being ignored. Use the `commThreads` parameter to set the number of threads to be used for communications between the clients and one or more backends.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0182A slapd was unable to start because of communication errors.

Severity: Immediate Action

Explanation: The LDAP server was unable to start any of the interfaces used for communication with clients.

System Action: The program ends.

Operator Response: Save the diagnostic information and contact the administrator.

Administrator Response: Check for other messages regarding errors during configuration. Correct the configuration file and restart the server.

GLD0183A The dllload function failed loading DLL *dll_name*; errno not available.

Severity: Immediate Action

Explanation: The LDAP server was unable to load the requested library. Due to the order of processing, errno information is not available. The most common reason for the error is that the specified DLL cannot be found in the libraries being searched.

System Action: The program continues. However, support provided by the DLL will not be available.

Operator Response: Contact the LDAP Administrator or see the Administrator Response. If the problem persists, contact the service representative.

Administrator Response: Correct the error and restart the server.

GLD0184I Connections allowed only on the secure port.

Severity: Information

Explanation: The LDAP server is allowing socket communications on the secure port only.

System Action: The program continues.

Administrator Response: None.

GLD0185I Connections allowed only on the nonsecure port.

Severity: Information

Explanation: The LDAP server is allowing socket communications on the nonsecure port only.

System Action: The program continues.

Administrator Response: None.

GLD0186I Connections allowed on both the secure and nonsecure port.

Severity: Information

Explanation: The LDAP server is allowing socket communications on both the secure and nonsecure ports.

System Action: The program continues.

Administrator Response: None.

GLD0187I Configuration parameter *configParameter* is being ignored since it was specified with the listen parameter.

Severity: Information

Explanation: When a valid `listen` parameter is specified in the configuration file, it takes precedence over what has been specified for the configuration parameter. The LDAP server will configure according to what was been specified for the `listen` parameter.

System Action: The program continues.

Administrator Response: Correct the configuration file by removing the configuration parameter.

GLD0188E Server is already listening on IP address: *ip_address* and port: *port_number*. Ignoring listen: *listen_arg*.

Severity: Eventual Action

Explanation: The configuration file has more than one `listen` parameter specified to listen on the same IP address and port number.

System Action: The program continues.

Administrator Response: Correct the `listen` parameter in the configuration file by specifying a different IP address or port number. Then restart the server.

GLD0189I Nonsecure communication is active for IP address: *ip_address*, nonsecure port: *port_number*.

Severity: Information

Explanation: The LDAP server has activated the specified IP address and port for nonsecure communication. An *ip_address* of 'IN_ADDRANY' indicates that the server is listening on all active interface addresses.

System Action: The program continues.

Administrator Response: None.

GLD0190I Secure communication is active for IP address: *ip_address*, secure port: *port_number*.

Severity: Information

Explanation: The LDAP server has activated the specified IP address and port for secure communication. An *ip_address* of 'IN_ADDRANY' indicates that the server is listening on all active interface addresses.

System Action: The program continues.

Administrator Response: None.

GLD0191A Nonsecure communication failed to activate for IP address: *ip_address*, port: *port_number*; reason code: *reason_code*.

Severity: Immediate Action

Explanation: The LDAP server encountered an error when attempting to activate nonsecure communication. An *ip_address* of 'IN_ADDRANY' indicates that the server attempted to listen on all active interfaces. Preceding messages should indicate the cause of the error. If there are no preceding messages or the reason codes documented in previous messages do not match the *reason_code* in this message, then the error can be one of the following:

- 19 Memory error in the server.
- -2001 Error encountered in socket() function.
- -2002 An application is already listening on the *ip_address* and *port_number* specified.

System Action: The program continues.

Operator Response: Correct the cause of the error.

GLD0192A Secure communication failed to activate for IP address: *ip_address*, port: *port_number*; reason code: *reason_code*.

Severity: Immediate Action

Explanation: The LDAP server encountered an error when attempting to activate secure communication. An *ip_address* of 'IN_ADDRANY' indicates that the server attempted to listen on all active interfaces. Preceding messages should indicate the cause of the error. If there are no preceding messages or the reason codes documented in previous messages do not match the *reason_code* in this message, then the error can be one of the following:

- 19 Memory error in the server.
- -2001 Error encountered in socket() function.
- -2002 An application is already listening on the *ip_address* and *port_number* specified.
- -3000 The `sslCertificate`, `sslKeyRingFile`, `sslKeyRingFilePW`, `sslKeyRingPWStashFile`, or `sslAuth` options are not valid or are not specified in the configuration file. There could be a problem finding or loading the SSL library.

System Action: The program continues.

Operator Response: Correct the cause of the error.

GLD0193E Debug level is not valid; reason code: *reason_code*, incorrect value '*bad_token*' detected at offset *offset* in string '*debug_string*'.

Severity: Eventual Action

Explanation: The LDAP server determined that the value specified for the debug level on either the **-d** option on the command line invocation of the program or a console command is not valid. The *reason_code* indicates why the value is not accepted. It can be:

- -1 Memory error in the server
- -2 Adjacent signs detected (for example, +-keyword)
- -3 Debug value is too large
- -4 Debug value is not a valid hexadecimal
- -5 Debug value is not a valid decimal or keyword
- -6 Off debug used with a sign (for example, +OFF, -OFF)
- -7 Debug value is only a sign (for example, +, -)
- -8 Debug value ends with a sign (for example, 8+, OFF-)

The token that caused the error is shown in the message along with its *offset* from the beginning of the *debug_string*. The debug level is ignored.

System Action: The program continues.

Operator Response: If debug is needed, restart the server with corrected command line options or specify a modify command from the operator's console to correctly specify the debug level.

Administrator Response: Correct the command line option and restart the server.

GLD0194I Successfully set debug level to *debug_level*.

Severity: Information

Explanation: The debug level was set to the specified value.

System Action: The program continues.

Administrator Response: None.

GLD0196A Attempt to accept() on IP address: *ip_address*, port *port_number* failed; errno *errno* (*errno_string*).

Severity: Immediate Action

Explanation: The LDAP server received an error from system function **accept()** on the specified IP address and port. Refer to *z/OS C/C++ Run-Time Library Reference*, SC28-1663 for an explanation of the *errno* returned.

System Action: The LDAP server stops listening on the specified IP address and port.

Operator Response: Ensure TCP/IP is operating correctly. If the problem was caused by TCP/IP being stopped, then shut down the LDAP server and restart it once TCP/IP is restarted. If the problem was caused by another error, then correct the problem and shut down the LDAP server and restart it once the problem is corrected. If the problem cannot be corrected, then save the diagnostic information and contact the system programmer.

Administrator Response: If the problem persists, contact the service representative.

GLD0197I The listen option in the configuration file is being overridden on the command line: *listen_arg*

Severity: Information

Explanation: The LDAP server has now been assigned to bind and listen based on the value supplied on the command line.

System Action: The program continues.

Administrator Response: None.

GLD0198I The -p option on the command line is being ignored.

Severity: Information

Explanation: The **-p** option is being ignored because there is at least one `listen` option specified in the configuration file or on the command line. The one or more `listen` options in the configuration file or on the command line will take precedence.

System Action: The program continues.

Administrator Response: None.

GLD0199I The -s option on the command line is being ignored.

Severity: Information

Explanation: The **-s** option is being ignored because there is at least one `listen` option specified in the configuration file or on the command line. The one or more `listen` options in the configuration file or on the command line will take precedence.

System Action: The program continues.

Administrator Response: None.

GLD0200E The value for the pcThreads option is not numeric. The default (*default_pc_threads*.) will be used.

Severity: Eventual Action

Explanation: The LDAP server determined that the value specified for the `pcThreads` option of the

configuration file is not numeric. The server will continue with the default value.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0201E The value for the pcThreads option is out of range (*min_threads*, *max_threads*). The default (*default_pc_threads*) will be used.

Severity: Eventual Action

Explanation: The LDAP server encountered an error when processing the `pcThreads` parameter of the configuration file. The server will continue with the default value.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0202I Program Call communication is active.

Severity: Information

Explanation: The LDAP server has activated support for program call communication.

System Action: The program continues.

Administrator Response: None.

GLD0203A The Program Call initialization failed; rc: *return_code*, reason code: *reason_code*.

Severity: Immediate Action

Explanation: The LDAP server encountered an error during initialization of the program call environment. Communication with the LDAP server using program calls is not enabled.

System Action: PC initialization is terminated. Initialization continues for all other requested communication interfaces. If no communication interface is successfully initialized, the program ends.

Operator Response: Inform the System Programmer of the error.

Administrator Response: Restart the server after the system programmer has resolved the problem.

Programmer Response: Use the return code and reason code information to correct the problem. Possible return codes are:

- 2 - another server has already enabled PC communications.
- 3 - cannot establish an ESTAEX exit. The reason code is the return code from the ESTAEX macro.
- 5 - cannot obtain a system linkage index. The reason code is the return code from the LXRES macro.
- 6 - cannot create a PC entry table. The reason code is the return code from the ETCRE macro.
- 7 - cannot connect the PC entry table to the linkage table. The reason code is the return code from the ETCON macro.
- 8 - cannot create a named token. The reason code is the return code from the IEANTCR macro.
- 9 - cannot make the address space non-swappable. The reason code is the return code from the SYSEVENT macro.
- 10 - there is insufficent memory available.

GLD0204E Additional listen configuration statements for Program Calls are ignored.

Severity: Eventual Action

Explanation: The LDAP server encountered multiple `listen` configuration statements for the program call environment. Only one is allowed. Additional program call `listen` configuration statements are ignored.

System Action: The program continues.

Operator Response: Inform the Administrator of the error.

Administrator Response: Correct the configuration file to specify only one `listen` statement for the program call environment.

GLD0205A Program Call communication failed to activate; return code: *return_code*

Severity: Immediate Action

Explanation: The LDAP server encountered an error when attempting to activate program call communication. Preceding messages should indicate the cause of the error. If there are no preceding messages or the return codes documented in previous messages do not match the *return_code* in this message, then the error can be one of the following:

- 19 Memory error in the server.
- -1005 Failed to initialize threads in the PC callable thread pool.
- -2013 Failed loading the PC Callable dll.
- -4000 Initialization of the PC Callable layer failed.
- -4002 Multiple `listen` PC Callable statements have been specified.

System Action: The program continues.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Examine other messages and correct any other reported errors. Restart the LDAP server.

GLD0206A Program Call (PC) support is not initialized because another server with PC support is already running.

Severity: Immediate Action

Explanation: There can be only one active LDAP server with program call (PC) support. Before initializing PC support, a new LDAP server checks that no active LDAP server on this system has already initialized PC support. If this is the case, PC support initialization in the new LDAP server is terminated.

System Action: PC initialization is terminated. Initialization continues for all other requested communication interfaces. If no communication interface is successfully initialized, the program ends.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: If it is necessary to have PC support running in the new LDAP server, first remove the `listen` record for PC support from the configuration of the active LDAP server. Then restart both that LDAP server and the new LDAP server. The problem can be avoided by ensuring that only one LDAP server is configured to provide PC support.

GLD0207I *backend_identifier* **manages the following suffixes:**

Severity: Information

Explanation: The LDAP server is reporting the suffixes managed by the backend.

System Action: The program continues.

Administrator Response: None.

GLD0208I Backend suffix: *backend_suffix*

Severity: Information

Explanation: The LDAP server is reporting the suffixes managed by the backend.

System Action: The program continues.

Administrator Response: None.

GLD0209I End of suffixes managed by *backend_identifier***.**

Severity: Information

Explanation: The LDAP server has completed the list of suffixes managed by the backend.

System Action: The program continues.

Administrator Response: None.

GLD0210I Modify command has been processed successfully: *modify_text*

Severity: Information

Explanation: A modify command has been processed successfully.

System Action: The program continues.

Administrator Response: None.

GLD0211A The value specified for the adminDN option is 'cn=Anybody'. This is not permitted.

Severity: Immediate Action

Explanation: The LDAP server encountered an error during processing of the `adminDN` option of the configuration file. The `adminDN` cannot be 'cn=Anybody'.

System Action: The program continues. If all required parameters are not set correctly, the configuration will fail and the server will not start. If the configuration parameter that failed is associated with a specific database backend, the server may start, but the backend may not be available. If a configuration parameter is ignored, the server may not operate as expected.

Administrator Response: Correct the configuration file and restart the server.

GLD0212E Cipher specification value is not valid; reason code: *reason_code***, incorrect value '***bad_token***' detected at offset** *offset* **in string '***cipher_string***'.**

Severity: Eventual Action

Explanation: The LDAP server determined that the value specified for the cipher is not valid. The `reason_code` indicates why the value is not accepted. It can be:

- -1 Memory error in the server
- -2 Adjacent signs detected (for example, +-keyword)
- -3 Cipher value is too large
- -4 Cipher value is not a valid hexadecimal
- -5 Cipher value is not a valid decimal or keyword
- -6 Off Cipher used with a sign (for example, +OFF, -OFF)
- -7 Cipher value is only a sign (for example, +, -)
- -8 Cipher value ends with a sign (for example, 8+, OFF-)
- -9 value specifies mask bits which are not allowed
- -10 an internal processing error was detected

The token that caused the error is shown in the message along with its *offset* from the beginning of the *cipher_string*. The cipher is ignored.

System Action: The program continues.

Operator Response: If a cipher specification is needed, restart the server with the cipher specification corrected.

Administrator Response: If the reason code indicates an internal processing error, then contact IBM support and provide the message text. Otherwise, correct the cipher specification.

GLD0213I Cipher specifications mask set to *cipher_mask***.**

Severity: Information

Explanation: The LDAP server has assigned the cipher specification mask to the specified value based on the value read from the configuration file and the values supported by SSL.

System Action: The program continues.

Operator Response: None.

GLD0214E SSL cipher lowered from the specified value '*cipher_mask***' to '***lower_cipher_mask***'**

Severity: Eventual Action

Explanation: The specified cipher value was not fully supported by SSL. A lower cipher value is used. This is normally caused by an incorrect specification of the cipher values or specification of a cipher value that is not allowed in the region of the world where the LDAP server is running.

System Action: The program continues. If a cipher specification is needed, restart the server with the cipher specification corrected.

Administrator Response: None.

GLD0218E The value specified (*value_specified***) for the serverEtherAddr option is not correct. The default value of the CPU model and serial number will be used.**

Severity: Eventual Action

Explanation: The LDAP server encountered an error during processing of the `serverEtherAddr` option in the configuration file. The default value will be used.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0219A Dynamic load of Kerberos DLL *dll_name* **failed. Errno:** *errno* **(***errno_string***).**

Severity: Immediate Action

Explanation: The server was unable to load the Kerberos DLL.

System Action: The program ends.

Operator Response: None.

Administrator Response: Check your Kerberos configuration.

GLD0220A Backend with name '*backend_name***' already exists.**

Severity: Immediate Action

Explanation: The LDAP server encountered an error processing the configuration file because the specified backend name is already in use by another backend. Note that `cdbm1` is a reserved name and therefore cannot be used in the configuration file. Also, backend names are not case sensitive. For example, `server1` and `SERVER1` are duplicate values.

System Action: The program ends.

Administrator Response: Change the duplicate name to a unique string or remove the name from the `database` line in the configuration file, and then restart the server.

GLD0221A Cannot configure more than one *backend_type* **backend.**

Severity: Immediate Action

Explanation: The LDAP server encountered an error processing the configuration file because a second database section of the type specified in the message was found. There can be at most one database section of this type in the configuration file.

System Action: The program ends.

Administrator Response: Correct the configuration file so that it contains at most one database section of this type and restart the server.

GLD0222E Replica server on *replica_name* **on port** *port_number* **does not contain support for replicating complex Modify DN operations.**

Severity: Eventual Action

Explanation: The replica server does not contain support for Modify DN operations with newSuperior, or IBMModifyDNRealignDNAttributesControl, or non-leaf Modify DN operations.

System Action: The program continues. Modify DN

operations with newSuperior, IBMModifyDNRealignDNAttributesControl, or non-leaf Modify DN operations will not be permitted.

Operator Response: None.

Administrator Response: Install the correct version of LDAP on the replica server or remove it from the replica collection.

GLD0223A Error *error_value* **reported by** *ldap_api* **querying replica server version on** *replica_name* **on port** *port_number***.**

Severity: Immediate Action

Explanation: An LDAP API call failed while querying the replica server version.

System Action: The program continues. Modify DN operations with newSuperior, IBMModifyDNRealignDNAttributesControl, or non-leaf Modify DN operations will not be permitted.

Operator Response: Save the diagnostic information and contact the system programmer.

Administrator Response: If the problem persists, contact the service representative.

GLD0224A One or more replica servers do not implement the Modify DN function.

Severity: Immediate Action

Explanation: One or more replica servers do not implement, or cannot be determined to implement, the requisite Modify DN function. However there is a committed Modify DN operation on the master server which cannot be replicated to one or more of the replicas. This state can result in continuing replication failures (until the situation is rectified) which will result in diverging directory contents between master server and one or more replica servers.

System Action: The program continues. The replication operation will not be performed.

Operator Response: None.

Administrator Response: Make sure that all replicas have the required level of the LDAP server.

GLD0225I Ignoring suffix: *backend_suffix* **defined for exop backend.**

Severity: Information

Explanation: The LDAP server encountered a suffix for a backend of type `exop` in the configuration file. An exop backend does not require a suffix so it will be ignored.

System Action: The program continues.

Administrator Response: None required. Correct the configuration file.

GLD0226E The 'attribute' parameter is no longer supported. All 'attribute' lines will be ignored.

Severity: Eventual Action

Explanation: The `attribute` parameter is now obsolete and is being ignored. Attribute definitions should be provided as documented for each server backend type. See the *z/OS Integrated Security Services LDAP Server Administration and Use* for more information.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0227E The 'objectclass' parameter is no longer supported. All 'objectclass' lines will be ignored.

Severity: Eventual Action

Explanation: The `objectclass` parameter is now obsolete and is being ignored. Object class definitions should be provided as documented for each server backend type. See *z/OS Integrated Security Services LDAP Server Administration and Use* for more information.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0228E The 'verifySchema' parameter is no longer supported. This line will be ignored.

Severity: Eventual Action

Explanation: The `verifySchema` parameter is now obsolete and is being ignored. The `verifySchema` parameter was only applicable for `attribute` and `objectclass` parameters, which are no longer supported. Note that schema loaded into the LDAP server for TDBM, GDBM, and SDBM is always verified, regardless of the presence of the `verifySchema` parameter.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0229E The activity log file could not be opened.

Severity: Eventual Action

Explanation: The LDAP server could not open the activity log file. This is due to an error in specifying the `logfile` configuration parameter.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0230A A self-signed SSL certificate cannot be validated because it is not in the SSL key database or key ring.

Severity: Immediate Action

Explanation: A self-signed certificate cannot be validated because it is not in the key database file or key ring.

System Action: The program continues. The request fails.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Add the self-signed certificate to the key database file or key ring.

GLD0231A The SSL Peer application sent a certificate for which the certification authority is not in the SSL key database or key ring.

Severity: Immediate Action

Explanation: The key database file or key ring does not contain a certificate for the certification authority.

System Action: The program continues. The request fails.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Obtain the certificate for the certification authority and add it to the key database file or key ring.

GLD0232A The SSL server certificate is not compatible with the negotiated cipher suite.

Severity: Immediate Action

Explanation: The certificate key is not compatible with the negotiated cipher suite. The server certificate must have an RSA key while the client certificate may have an RSA or DSA key. This error can also occur if the client certificate has a DSA key but the server does not support DSA keys, the server key usage certificate extension does not allow key encipherment, or the client key usage certificate extension does not allow digital signature. For the 40-bit export ciphers, the server key usage certificate extension must allow digital signature.

System Action: The program continues. The request fails.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Specify a certificate with the appropriate key type and key usage.

GLD0233A SSL detected an error while validating a certificate.

Severity: Immediate Action

Explanation: An error is detected while validating a certificate. This error can occur if a root CA certificate is not found in the key database file or key ring or if the certificate is not marked as a trusted certificate.

System Action: The program continues. The request fails.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Verify that the root CA certificate is in the key database or key ring and is marked as trusted. Check all certificates in the certification chain and verify that they are trusted and are not expired.

GLD0234A A memory allocation error occurred in SSL processing.

Severity: Immediate Action

Explanation: The LDAP server is unable to allocate the necessary storage for SSL processing.

System Action: Communication interfaces that require SSL will not be activated. Communication interfaces that support both secure and non-secure communication will be activated for non-secure communication only. If no interfaces are activated then the program will end.

Operator Response: Increase the storage for the LDAP server and restart the server.

GLD0235A The maximum connections limit could not be set to '*new_max_connection_limit*'. The setrlimit() call failed with errno='*errno*' (*errno_string*). The limit remains unchanged at '*existing_max_connection_limit*'.

Severity: Immediate Action

Explanation: The LDAP server was unable to change the number of connections allowed to the limit specified by the `maxconnections` parameter in the configuration file. The limit is affected by the number of file descriptors currently open for the process and the system limit of 65,535.

System Action: The program continues with the existing maximum connections value.

Administrator Response: Determine the reason for the error. The `errno` value set by setrlimit() is provided to assist in identifying the error. Correct the

maxconnections value in the configuration file and start the server again.

GLD0236I IP address: *ip_address*, port *port_number* has been stopped.

Severity: Information

Explanation: The LDAP server has stopped communication on the specified IP address and port. No new connections will be accepted on the specified interface.

System Action: The LDAP server stops communicating on the specified IP address and port.

Operator Response: If communication is desired on the specified interface, then stop the LDAP server, if it is still running. Restart the LDAP server once it is stopped.

Administrator Response: None.

GLD0237I Program Call Interface has been stopped.

Severity: Information

Explanation: The LDAP server has stopped communication on the program call interface. No new connections will be accepted on the specified interface.

System Action: The LDAP server stops communicating on the specified interface.

Operator Response: If communication is desired on the specified interface, then stop the LDAP server, if it is still running. Restart the LDAP server once it is stopped.

Administrator Response: None.

GLD0238A Backend type (*type*) and DLL do not match.

Severity: Immediate Action

Explanation: The LDAP server encountered an error processing the configuration file. The specified database type and the DLL to load do not match. One possible reason is a typographical error in the configuration file.

System Action: The program continues through configuration processing but the configuration will fail and the server will not start.

Administrator Response: Check the database lines in the configuration file for the specified backend type and verify the correct DLL is being loaded. Correct the configuration file and restart the server. If the problem persists, contact the service representative.

GLD0239A The ibmdirectoryversion attribute is not found in the Root DSE of the replica server on *replica_name* on port *port_number*.

Severity: Immediate Action

Explanation: It is not possible to determine if the replica server contains support for Modify DN operations with newSuperior, IBMModifyDNRealignDNAttributesControl, or non-leaf Modify DN operations or support for **ibm-entryuuid** attributes.

System Action: The program continues. The **ibm-entryuuid** attributes will not be replicated. Modify DN operations with newSuperior, IBMModifyDNRealignDNAttributesControl, or non-leaf Modify DN operations will not be permitted.

Administrator Response: Install a conformant version of LDAP on the replica server or remove it from the replica collection.

GLD0240A The ibmdirectoryversion attribute has no value in the Root DSE of the replica server on *replica_name* on port *port_number*.

Severity: Immediate Action

Explanation: It is not possible to determine if the replica server contains support for Modify DN operations with newSuperior, IBMModifyDNRealignDNAttributesControl, or non-leaf Modify DN operations, or support for **ibm-entryuuid** attributes.

System Action: The program continues. The **ibm-entryuuid** attributes will not be replicated. Modify DN operations with newSuperior, IBMModifyDNRealignDNAttributesControl, or non-leaf Modify DN operations will not be permitted.

Administrator Response: Install a conformant version of LDAP on the replica server or remove it from the replica collection.

GLD0241A The replica server on *replica_name* on port *port_number* does not support the ibm-entryuuid attribute.

Severity: Immediate Action

Explanation: The replica server does not have the necessary code level to support the **ibm-entryuuid** attribute.

System Action: Replication will not occur to that system until the problem is corrected.

Administrator Response: Install a conformant version of LDAP on the replica server or remove it from the replica collection.

GLD0242I Approaching maximum number of concurrent client connections, currently using *current_connections* out of *max_connections*.

Severity: Information

Explanation: The LDAP server is almost at the maximum number of concurrent client connections that it can currently support. When the number of concurrent client connections reaches the maximum, new client applications attempting to connect to the LDAP server will be rejected with a network error. These network errors can be caused by client applications not unbinding when they are finished communicating with the LDAP server.

System Action: The program continues. When *current_connections* equals *max_connections*, additional client applications cannot connect to the LDAP server. This message will be issued at most once a minute for a limit of 60 times when this condition exists. Although this message may not be issued after the 60 times displayed on the console the condition may still exist.

Administrator Response: If `maxconnections` is set in the LDAP configuration file, increase its value. Check that the increased value of `maxconnections` can be supported by obtaining the settings of the `MAXFILEPROC` statement and `MAXSOCKETS` on the `NETWORK` statement in `BPXPRMxx`. If `maxconnections` is not set in the LDAP configuration file, check that the settings of the `MAXFILEPROC` statement and `MAXSOCKETS` on the `NETWORK` statement in `BPXPRMxx` are set to a sufficient setting. After making updates to the LDAP configuration file or to the statements in `BPXPRMxx`, it is necessary to restart the LDAP server to put these changes into effect. Ensure that client applications disconnect when they are finished making requests to the LDAP server.

GLD0243I The number of concurrent client connections is now at *current_connections*, below the warning threshold of *threshold_connections*.

Severity: Information

Explanation: The LDAP server is now below the warning threshold of concurrent client connections. Warning messages maybe issued once again, if the number of concurrent client exceeds the warning threshold.

System Action: The program continues.

Administrator Response: If this message is repeatedly displayed it means that the maximum number of file descriptors available to the LDAP server maybe too low. In this case, it may be desirable to increase the number of concurrent connections that the LDAP server can support, by changing the settings of `maxconnections` in the LDAP configuration file. Check that the values `MAXSOCKETS` on the `NETWORK` statement to ensure that the new maximum number of *concurrent_connections* is permitted. If you change any of these values, you must stop and restart the LDAP Server to put the changes into effect.

GLD0244I Change logging is enabled
Logging started status (0 = off, 1 = on): *flag*
Limit in seconds on age of change log entries (0 = no limit): *limit*
Limit on the number of change log entries (0 = no limit): *limit*
Current number of change log entries: *number*
First change number in use: *changenumber*
Last change number in use: *changenumber*

Severity: Information

Explanation: This message is issued to indicate that change logging is enabled for the LDAP server and to specify the status of the change log. Note that change log entries can be read, modified and deleted even when change logging is off; however, new entries are only added to the change log when change logging is on.

System Action: The program continues.

Administrator Response: None.

GLD0245A Change log configuration has failed and change logging is not enabled.

Severity: Immediate Action

Explanation: Due to an error during configuration of the change log, the change log is not started. No operations on the change log can take place. In particular, new change log entries are not created when other directory entries are changed.

System Action: The program continues.

Administrator Response: Stop the LDAP server if it should not run without change logging enabled. Use the information in any previous messages and debug output to resolve the problem. Then start the server again.

GLD0246A A suffix *backend_suffix* is detected that overlaps suffix *changelog_suffix*.

Severity: Immediate Action

Explanation: A suffix that overlaps the change log suffix is detected during configuration of the LDAP server. This is not allowed when change logging is enabled.

System Action: The program ends.

Administrator Response: Either change the overlapping suffix or remove it from the configuration file. Then start the server again.

GLD0247A *program_name*: **line** *line_number*: **incorrect configuration line: keyword** '*keyword*' **is not allowed.**

Severity: Immediate Action

Explanation: The LDAP server encountered an error processing the configuration file because the specified line contained a keyword that is not allowed in this section of the configuration file.

System Action: The program ends.

Operator Response: None.

Administrator Response: Correct the configuration file and start the server again.

GLD0248A **Unable to start change log trim thread, rc=***return_code***.**

Severity: Immediate Action

Explanation: An attempt to start the change log trim thread failed with the specified return code.

System Action: The program ends.

Operator Response: Increase the region size and restart the server.

Administrator Response: None.

GLD0249A **Change log trim has ended.**

Severity: Immediate Action

Explanation: The change log trim thread trims the change log to prevent it from growing too large. The change log will not be trimmed.

System Action: None.

Operator Response: Restart the server.

Administrator Response: None.

GLD0250E **The value (***db2terminate_conf_value***) specified for the db2terminate option is not valid. The default (***db2terminate_default***) will be used.**

Severity: Eventual Action

Explanation: The LDAP server encountered an error during processing of the `db2terminate` option of the configuration file. The default value of restore will be used.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0251E **DB2 termination detected, server is stopping.**

Severity: Eventual Action

Explanation: The LDAP Server has detected a DB2 termination, and the `db2terminate` configuration option is set to 'terminate'.

System Action: The program terminates.

Operator Response: Restart DB2 and then restart the LDAP server.

Administrator Response: None.

GLD0252E **DB2 termination detected, database access unavailable.**

Severity: Eventual Action

Explanation: The LDAP server has detected DB2 termination and the `db2terminate` configuration option is set to 'recover'. The LDAP server will re-connect to DB2 when DB2 becomes available.

System Action: The program continues. Any client requests to access data stored in DB2 will fail.

Operator Response: Restart DB2.

Administrator Response: None.

GLD0253I **DB2 restart detected, database access available.**

Severity: Information

Explanation: The LDAP server has detected a restart of DB2. Database access is available.

System Action: The program continues.

Operator Response: None.

Administrator Response: None.

GLD0254A **The LDAP server encountered an error initializing the DB2 monitor. return code =** *return_code***.**

Severity: Immediate Action

Explanation: The LDAP server encountered an error during db2 monitor configuration. The program cannot start. See other messages regarding errors.

System Action: The program ends.

Operator Response: Contact the LDAP Administrator or see the Administrator Response. If the problem persists, contact the service representative.

Administrator Response: Ensure adequate memory for the program. Examine other messages and correct any other reported errors. Restart the LDAP server.

GLD0255A The LDAP server encountered an error during replication configuration. return code = *return_code*.

Severity: Immediate Action

Explanation: The LDAP server encountered an error during replication configuration. The program cannot start. See other messages regarding errors.

System Action: The program ends.

Operator Response: Contact the LDAP Administrator or see the Administrator Response. If the problem persists, contact the service representative.

Administrator Response: Ensure adequate memory for the program. Examine other messages and correct any other reported errors. Restart the LDAP server.

GLD0256E The value for the *option* option is not numeric. The value *value* will be used.

Severity: Eventual Action

Explanation: The LDAP server determined that the value for the specified option in the configuration file was not numeric. The server will continue with the value specified.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0257E The value for the *option* option is out of range (*minimum_value*, *maximum_value*). The value *value* will be used.

Severity: Eventual Action

Explanation: The LDAP server determined that the numeric value for the specified option in the configuration file was not within the range of permitted values. The server will continue with the value specified.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD0258E The local hostname of the LDAP server could not be determined.

Severity: Eventual Action

Explanation: The LDAP server could not properly resolve its own fully qualified hostname. The server will continue, however, operations involving extended operations that proxy to another LDAP server to perform LDAP operations may fail.

System Action: The program continues.

Operator Response: Contact the Administrator or see the Administrator Response.

Administrator Response: Ensure that the LDAP server has access to a DNS so that the fully qualified local hostname can be determined.

GLD0259A The LDAP server cannot be configured as both a read only replication server and a peer replication server.

Severity: Immediate Action

Explanation: The LDAP server cannot be configured as a read only replication server and as a peer replication server. Both `peerServerDN` and the `masterServer` configuration option (either `masterServer`, `masterServerDN`, or `masterServerPW`) are present in the configuration file.

System Action: The program ends.

Operator Response: None.

Administrator Response: Correct one of the configuration file values and try again.

GLD0260E *type_of_operation* replication operation to server *hostname_and_port* has failed. Change number=*change_number* Return code=*return_code* Reason code=*reason_code* Additional Information=*additional_info*.

Severity: Eventual Action

Explanation: The LDAP server has detected that a replication server has sent or received conflicting information and replication to that server has stalled.

System Action: The program continues. However, the LDAP replication server is stalled.

Administrator Response: Check all replication servers for out of sync data. Refer to the Chapter 23, “Replication,” on page 289 for out of sync recovery information.

GLD0261E Add replication operation found that entry *dn* already exists on replication server *server_and_port*. Existing entry has been replaced.

Severity: Eventual Action

Explanation: The LDAP server has detected that the entry previously existed. The existing entry has been deleted and an add for the entry has been sent to the replication server.

System Action: The program continues.

Operator Response: None.

Administrator Response: Verify that the added entry is correct on all replication servers and that all replication servers are in sync. Refer to the Chapter 23, “Replication,” on page 289 for out of sync recovery information.

GLD0262I Delete replication operation found that entry *dn* does not exist on replication server *server_and_port*. Delete replication operation ignored.

Severity: Information

Explanation: The LDAP server has detected that the entry does not exist. The replication delete operation will be ignored.

System Action: The program continues.

Administrator Response: Verify that all replication servers are in sync. Refer to the Chapter 23, "Replication," on page 289 for out of sync recovery information.

GLD0263I Slapd is ready for requests (Maintenance mode).

Severity: Information

Explanation: The LDAP server is in maintenance mode and is ready for requests. Requests will only be accepted from the `masterServerDN` or `peerServerDN`. Maintenance mode is the LDAP server setup mode for peer replication.

System Action: The program continues.

Operator Response: None.

Administrator Response: None.

GLD0264I Slapd Maintenance mode is starting.

Severity: Information.

Explanation: A console modify command was received, starting maintenance mode. Maintenance mode is the LDAP server setup mode for peer replication.

System Action: The program continues.

Operator Response: None.

Administrator Response: None.

GLD0265I Slapd Maintenance mode is ending.

Severity: Information.

Explanation: A console modify command was received, ending maintenance mode. Maintenance mode is the LDAP server setup mode for peer replication.

System Action: The program continues.

Operator Response: None.

Administrator Response: None.

GLD0266A *option value1* in *backend1* backend does not match *option value2* in *backend2* backend.

Severity: Immediate Action

Explanation: The LDAP server encountered an error during configuration. Configuration of a server with multiple TDBM backends encountered multiple values for `peerServerDN` or `masterServerDN` that did not match. All values for `peerServerDN` or `masterServerDN` must be the same.

System Action: The program ends.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Correct the configuration file and restart the server.

GLD2086E Encrypt all passwords that are presently in CLEAR format (yes/no)?

Severity: Eventual Action

Explanation: A db2pwden utility replaces clear text passwords with encrypted passwords. Prompt a user with this message to give her/him a chance to change their mind. If the response was 'yes' (or 'y' or 'Y'), the program will continue, otherwise it will stop.

System Action: The program continues or ends based upon input.

Operator Response: None.

Administrator Response: None

GLD2091A No base is defined.

Severity: Immediate Action

Explanation: A base must be defined either on the command line or through the **LDAP_BASEDN** environment variable.

System Action: The program ends.

Operator Response: None.

Administrator Response: Define a base and try the program again.

GLD2092A db2pwden ends without encrypting passwords.

Severity: Immediate Action

Explanation: The user changed their mind or an error occurred which caused the **db2pwden** program to end without encrypting passwords.

System Action: The program ends.

Operator Response: None.

Administrator Response: If the user wants to try the program again, correct any errors identified and try again.

GLD2100A Memory allocation failed.

Severity: Immediate Action

Explanation: The progam has used all of the available storage.

System Action: The program ends.

Operator Response: None.

Administrator Response: Increase the region size and try again.

GLD2101A The only supported mechanisms are EXTERNAL, GSSAPI, CRAM-MD5, and DIGEST-MD5.

Severity: Immediate Action

Explanation: The progam does not support the requested authentication mechanism.

System Action: The program ends.

Operator Response: None.

Administrator Response: Specify EXTERNAL, GSSAPI, CRAM-MD5, or DIGEST-MD5.

GLD2102A Scope should be base, one, or sub.

Severity: Immediate Action

Explanation: The progam does not support the requested scope mechanism.

System Action: The program ends.

Operator Response: None.

Administrator Response: Specify either base, one or sub.

GLD2103A Error *error_code* and reason *reason_code* reported by SSL Client initialization.

Severity: Immediate Action

Explanation: The LDAP utility was unable to complete initialization required for secure communications.

System Action: The program terminates.

Operator Response: Contact the service representative.

GLD2104A Error reported by LDAP client initialization.

Severity: Immediate Action

Explanation: The LDAP utility was unable to complete initialization required for communications.

System Action: The program terminates.

Operator Response: Contact the service representative.

GLD2105A Error *error_code* reported binding to LDAP server.

Severity: Immediate Action

Explanation: The LDAP utility was unable to bind to the server.

System Action: The program terminates.

Operator Response: Contact the service representative.

GLD2106A Error *error_code* reported modifying DN *distinguished_name*.

Severity: Immediate Action

Explanation: The LDAP utility was unable to add the **ibm-entryuuid** attribute to the LDAP entry.

System Action: The program terminates.

Operator Response: Contact the service representative.

GLD2107A Error *error_code* reported parsing LDAP results.

Severity: Immediate Action

Explanation: The LDAP utility was unable to add the **ibm-entryuuid** to the LDAP entry.

System Action: The program terminates.

Operator Response: Contact the service representative.

GLD2108A Error *error_code* reported by ldap_search.

Severity: Immediate Action

Explanation: The LDAP search failed.

System Action: The program terminates.

Operator Response: Contact the service representative.

GLD2109A Error *error_code* reported by ldap_get_dn.

Severity: Immediate Action

Explanation: The LDAP utility was unable to get the DN.

System Action: The program terminates.

Operator Response: Contact the service representative.

GLD2110A Error *error_code* reported by ldap_first_entry.

Severity: Immediate Action

Explanation: An LDAP internal error occurred.

System Action: The program terminates.

Operator Response: Contact the service representative.

GLD2111I ldapadduids added ibm-entryuuids to *number_matched* entries.

Severity: Information

Explanation: The **ldapadduuids** utility added ibm-entryuuids to the specified number of entries.

System Action: None.

Operator Response: None.

GLD2112I *command_name* processed *number_matched* entries.

Severity: Information

Explanation: The utility processed the specified number of entries.

System Action: None.

Operator Response: None.

GLD2113A A bind DN is required.

Severity: Immediate Action

Explanation: The LDAP utility will not function unless a bind DN is specified.

System Action: The program terminates.

Operator Response: Specify a bind DN.

GLD2114A A base DN must be specified.

Severity: Immediate Action

Explanation: The LDAP utility will not function unless a base DN is specified.

System Action: The program terminates.

Operator Response: Specify a base DN either with the **-b** option or with the **LDAP_BASEDN** environment variable.

GLD2116A A user name is required when doing a DIGEST-MD5 bind.

Severity: Immediate Action

Explanation: The LDAP utility will not function unless a user name is specified.

System Action: The program terminates.

Operator Response: Specify a user name with the **-U** option.

GLD2117A Debug value is not valid.

Severity: Immediate Action

Explanation: The LDAP utility will not function because the specified debug value is not valid.

System Action: The program terminates.

Operator Response: Specify a valid debug value with the **-d** option.

GLD2118A Incorrect DN syntax.

Severity: Immediate Action

Explanation: The LDAP utility was unable to bind to the server.

System Action: The program terminates.

Operator Response: Correct the DN syntax.

GLD2119A Incorrect LDAP server name or LDAP server is not available.

Severity: Immediate Action

Explanation: The LDAP utility was unable to bind to the server.

System Action: The program terminates.

Operator Response: Specify the correct LDAP server name or start the LDAP server.

GLD2120A Credentials are not valid for the specified LDAP server.

Severity: Immediate Action

Explanation: The LDAP utility was unable to bind to the LDAP server because the specified credentials were not accepted by the LDAP server.

System Action: The program terminates.

Operator Response: Specify the correct credentials for the LDAP server name.

GLD2121A Base DN not found on LDAP server.

Severity: Immediate Action

Explanation: The LDAP utility was unable to perform the operation because the specified base DN was not found by the LDAP server.

System Action: The program terminates.

Operator Response: Specify a base DN that exists on the LDAP server.

GLD2122A Bind to LDAP server failed, DN not found and default referral not permitted.

Severity: Immediate Action

Explanation: The LDAP utility was unable to bind to the LDAP server because the specified bind DN was not found by the LDAP server and the use of default referral is not permitted.

System Action: The program terminates.

Operator Response: Specify a DN that exists on the LDAP server.

GLD2123A Object not found on LDAP server.

Severity: Immediate Action

Explanation: The LDAP server was unable to find an object that matched the specification and that did not have an **ibm-entryuuid** attribute.

System Action: The program terminates.

Operator Response: Use **ldap_search** to make sure the object you are attempting to modify exists on the LDAP server. Specify **ibm-entryuuid** as the attribute **ldap_search** should return.

GLD2124A LDAP server unwilling to perform specified operation.

Severity: Immediate Action

Explanation: The LDAP server is not permitted to perform the operation.

System Action: The program terminates.

Operator Response: Check the options passed to the utility.

GLD2125A Error in search filter.

Severity: Immediate Action

Explanation: The search filter is not formed with the correct syntax or there is an incorrect character in the specification.

System Action: The program terminates.

Operator Response: Check the search filter.

TDBM messages (3000)

GLD3001I *time command_name*: *number* **entries have been processed.**

Severity: Information

Explanation: The program has processed the specified number of entries. Note that the program might have encountered errors during this processing. If so, processing of some entries may not have completed successfully. Additional messages are issued to indicate these errors.

System Action: The program continues.

Administrator Response: None.

GLD3002I *command_name* **has completed successfully.**

Severity: Information

Explanation: The command has ended without encountering any errors.

System Action: The program ends.

Administrator Response: None.

GLD3003I *command_name* **has failed.**

Severity: Information

Explanation: The command has ended after encountering an error.

System Action: The program ends.

Administrator Response: Use the information in any error messages issued by the command to fix the problem. Then try the command again.

GLD3004I **The check step of** *command_name* **has completed successfully.**

Severity: Information

Explanation: The check step of the **ldif2tdbm** program has ended without encountering any errors.

System Action: The program continues to process other requested steps.

Administrator Response: None.

GLD3005I **The check step of** *command_name* **has failed.**

Severity: Information

Explanation: The check step of the **ldif2tdbm** program has ended after encountering errors.

System Action: The program ends.

Administrator Response: Use the information in any error messages issued by the command to fix the problem. Then try the command again.

GLD3006I **The prepare step of** *command_name* **has completed successfully.**

Severity: Information

Explanation: The prepare step of the **ldif2tdbm** program has ended without encountering any errors.

System Action: The program continues to process other requested steps.

Administrator Response: None.

GLD3007I **The prepare step of** *command_name* **has failed.**

Severity: Information

Explanation: The prepare step of the **ldif2tdbm** program has ended after encountering errors. Existing data in the output files from a previous invocation of the prepare step have been deleted.

System Action: The program continues to process the check step if it has been requested. Otherwise, the program ends.

Administrator Response: Use the information in any error messages issued by the command to fix the problem. Then try the command again.

GLD3008I **The schema modify part of the load step of** *command_name* **has completed successfully.**

Severity: Information

Explanation: If one or more schema input files are specified with the **-s** or **-v** options, the first part of the load step of the **ldif2tdbm** program is to modify the schema in the database, using the schema input files. The schema in the database has been successfully updated.

System Action: The program continues to process the load step.

Administrator Response: None.

GLD3009I **The load step of** *command_name* **has successfully submitted the DB2 load utility jobs.**

Severity: Information

Explanation: The load step of the **ldif2tdbm** program submits the DB2 Load utility jobs created during the prepare step to load the new entries into the database. The load jobs have been successfully submitted. **Note:** This message does not indicate that the load jobs have terminated successfully. The processing of the load jobs by DB2 is outside the scope of the **ldif2tdbm** program. You must review the output generated by each load job to determine if it was successful.

System Action: The program ends.

Administrator Response: Review the output of each load to determine if it was successful. If not, use the information in the description of the **ldif2tdbm** command in *z/OS Integrated Security Services LDAP Server Administration and Use Guide* to determine how to proceed. **Note:** Do not run the **ldif2tdbm** program again because this can add duplicate data to the database.

GLD3010I **The load step of** *command_name* **has failed.**

Severity: Information

Explanation: The load step of the **ldif2tdbm** program has ended after encountering errors. However, if one or more schema input files were specified with the **-s** or **-v** options, the first part of the load step, which modifies the schema in the database, may have succeeded. If so, a message indicating this will precede this message.

System Action: The program ends.

Administrator Response: Use the information in any error messages issued by the command to fix the problem and determine how to proceed. If an error message was issued indicating a failure while submitting JCL, do not run the **ldif2tdbm** program again because this can add duplicate data to the database. Instead, use the information in the description of the **ldif2tdbm** command in *z/OS Integrated Security Services LDAP Server Administration and Use Guide* to determine how to proceed. If the JCL message was not issued, try the **ldif2tdbm** command again. Do not specify the **-s** or **-v** options to modify the schema if that part of the load step was successful.

GLD3011I **The** *command_name* **status file** *file_name* **cannot be written.**

Severity: Information

Explanation: The **ldif2tdbm** was not able to write the new status file after completing processing of the requested steps. The current status file may have been deleted. All other output files, such as the load and JCL datasets produced by the prepare step, have not been affected. This will likely result in warning messages the next time that **ldif2tdbm** is invoked for this output file.

System Action: The program ends.

Administrator Response: Use the information in the error messages issued by the command to fix the problem. If the error messages indicate that some processing step failed, run the command again. If **ldif2tdbm** issues warning messages and a prompt, be sure that all the conditions in the warning messages are acceptable before responding to continue processing.

GLD3012I *command_name* **has terminated because there are no entries to process.**

Severity: Information

Explanation: There are no entries for the program to process. For the **ldif2tdbm** program, there are no entries in the LDIF input files. For the **tdbm2ldif** program, there are no entries in the database.

System Action: The program ends.

Administrator Response: For **ldif2tdbm**, ensure that the LDIF input files contain entries to add to the database. For **tdbm2ldif**, check that the database contains entries to unload. Then try the program again.

GLD3013A *program_name***: line** *line_number***: incorrect configuration line:** *option* **takes parameters** *values***.**

Severity: Immediate Action

Explanation: The TDBM or GDBM backend cannot be configured because of the specified error in the configuration file.

System Action: The program continues. If all required parameters are not set correctly, the configuration will fail.

Administrator Response: Correct the configuration file and try again.

GLD3015A *program_name***: line** *line_number***: incorrect configuration line: unrecognized keyword.**

Severity: Immediate Action

Explanation: The TDBM or GDBM backend cannot be configured because the specified line in the configuration file contains an unrecognized keyword.

System Action: The program continues. If all required parameters are not set correctly, the configuration will fail and the server will not start. If the configuration parameter that failed is associated with a specific database backend, the server may start, but the backend may not be available. If a configuration parameter is ignored, the server may not operate as expected.

Administrator Response: Correct the configuration file and try again.

GLD3016A *keyword* **parameter is missing from configuration file.**

Severity: Immediate Action

Explanation: The TDBM or GDBM backend cannot be configured because the configuration file is missing a required parameter.

System Action: The program continues. If all required parameters are not set correctly, the configuration will fail and the server will not start. If the configuration parameter that failed is associated with a specific database backend, the server may start, but the backend may not be available. If a configuration parameter is ignored, the server may not operate as expected.

Administrator Response: Correct the configuration file and try again.

GLD3017A Unable to connect to the database, rc =*return_code*.

Severity: Immediate Action

Explanation: The LDAP server or LDAP utility cannot connect to DB2.

System Action: The program continues. The TDBM or GDBM backend ends.

Operator Response: Ensure that DB2 is started and running properly.

GLD3018A The dsnaoini CLI initialization file was not specified.

Severity: Immediate Action

Explanation: The LDAP server or LDAP utility cannot complete initialization of the DB2 interface because `dsnaoini` was not specified in the slapd.conf file, there is no DD specified for DSNAOINI in the JCL, or the DSNAOINI environment variable is not set.

System Action: The program continues. The TDBM or GDBM backend ends.

Administrator Response: Specify `dsnaoini` in the slapd.conf file, the DD DSNAOINI in the JCL, or set the DSNAOINI environment variable.

GLD3019E Entry '*distinguished_name*' already exists.

Severity: Eventual Action

Explanation: The request cannot be completed because an attempt is being made to add an entry that already exists in the database.

System Action: The program continues. The request fails.

Administrator Response: Correct the request and try again.

GLD3020E Entry '*distinguished_name*' violates the schema definition.

Severity: Eventual Action

Explanation: The request cannot be completed because an attempt is being made to add an entry that does not match the schema definition.

System Action: The program continues. The request fails.

Administrator Response: Correct the request and try again.

GLD3021E Parent entry does not exist for entry '*distinguished_name*'.

Severity: Eventual Action

Explanation: The request cannot be completed because no parent entry exists for the entry being added.

System Action: The program continues. The request fails.

Administrator Response: Correct the request and try again.

GLD3033I The LDAP server will operate in multi-server mode.

Severity: Information

Explanation: The LDAP server or LDAP utility will operate in multi-server mode. Multiple concurrent servers may be running on a given z/OS image, and multiple concurrent servers may be running on multiple z/OS images in a Parallel Sysplex, all of which use the same LDAP DB2 database. Note that no replication may be performed by any of these servers. If replication is desired, only one server instance may be running which uses a given LDAP DB2 database, and the server must be configured to run in single-server mode.

System Action: The program continues.

Administrator Response: None.

GLD3036E Multiserver keyword argument is 'n' or 'N', but the LDAP server will operate in multi-server mode.

Severity: Eventual Action

Explanation: The `multiserver` keyword argument found in the server configuration file was either 'n' or 'N'. However, because both `sysplexServerName` and `sysplexGroupName` keywords have valid arguments, the LDAP server or LDAP utility must operate in multi-server mode.

System Action: The program continues.

Administrator Response: Correct the configuration file and restart the server.

GLD3039A The LDAP program found no TDBM database.

Severity: Immediate Action

Explanation: The configuration file does not contain any TDBM database specifications. The program cannot continue because it cannot find the appropriate database with which to work.

System Action: The program ends.

Administrator Response: Ensure that the database needed by the program is specified in the configuration file. Then try the program again.

GLD3040A Overlapping TDBM backend suffixes found in configuration file: suffixes '*first_overlapping_suffix*' and '*second_overlapping_suffix*' overlap.

Severity: Immediate Action

Explanation: The presence of overlapping suffixes was detected in the server configuration file for a TDBM backend. Overlapping suffixes (suffixes for which the hierarchy is identical to the extent of the shorter of the two) may not be specified. Eliminate the overlap in suffixes and restart the server.

System Action: The program continues. Backends that configure successfully will be available. If no backends configure successfully, an additional message will appear and the program will end.

Administrator Response: Eliminate the overlap in TDBM suffixes and restart the server.

GLD3041E Parent of new entry is a referral object. Cannot add new entry '*distinguished_name*'.

Severity: Eventual Action

Explanation: The request cannot be completed because an entry cannot be added directly below a referral object. The entry must be added to the namespace below the actual entry which is referenced by the referral object.

System Action: The program continues. The request fails.

Administrator Response: Direct the request to the correct location in the namespace and try again.

GLD3042A The LDAP program encountered an error during configuration.

Severity: Immediate Action

Explanation: An LDAP program encountered an error while processing the configuration file. The program cannot continue. See additional messages for more information about the error encountered.

System Action: The program ends.

Administrator Response: Examine the additional messages. Correct the configuration file and try the program again.

GLD3043A The LDAP program found incomplete database information.

Severity: Immediate Action

Explanation: An LDAP program encountered an empty list of TDBM-specific information. The program cannot continue.

System Action: The program continues. Backends that configure successfully will be available. If no backends configure successfully, an additional message will appear and the program will end.

Administrator Response: Correct the configuration file and restart the server.

GLD3045A The LDAP program does not support this encryption method.

Severity: Immediate Action

Explanation: The LDAP configuration file contains the keyword `pwEncryption` with an incorrect value. Correct values are **none**, **crypt**, **MD5**, **SHA**, or **DES**.

System Action: The program continues. Backends that configure successfully will be available. If no backends configure successfully, an additional message will appear and the program will end.

Administrator Response: Correct the configuration file and restart the server.

GLD3046A OCSF setup failed, encryption method '*method*' is not available.

Severity: Immediate Action

Explanation: The `pwEncryption` value specified in the configuration file requires the OCSF product to be installed and available on your system.

System Action: The program continues. Backends that configure successfully will be available. If no backends configure successfully, an additional message will appear and the program will end.

Administrator Response: Verify OCSF setup and try again or change the `pwEncryption` value in the configuration file to a method that does not need OCSF (**none** or **crypt**).

GLD3047E OCSF setup failed, but pwEncryption method '*method*' is available.

Severity: Eventual Action

Explanation: The `pwEncryption` value specified in the configuration file does not require the OCSF product to

be installed and available on your system, but any value already encrypted in MD5, SHA, or DES will not compare correctly on a bind.

System Action: The program continues.

Administrator Response: If OCSF is needed, verify OCSF setup and then stop and start the LDAP program again.

GLD3048A DES key label not available, encryption method '*method*' is not available.

Severity: Immediate Action

Explanation: The `pwEncryption` value specified in the configuration file requires the OCSF product and the ICSF product to be installed and available on your system. It also requires a valid CKDS and the key corresponding to the DES key label in the configuration file to be available on your system.

System Action: The program continues. Backends that configure successfully will be available. If no backends configure successfully, an additional message will appear and the program will end.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Verify OCSF, ICSF, and CKDS setup and try again or change the `pwEncryption` value in the configuration file to a method that does not need OCSF, ICSF, or CKDS.

GLD3049A The DES key label specified with pwEncryption in the configuration file is too long.

Severity: Immediate Action

Explanation: The DES key label specified with `pwEncryption` in the configuration file can be a maximum of 64 characters long.

System Action: The program continues. Backends that configure successfully will be available. If no backends configure successfully, an additional message will appear and the program will end.

Administrator Response: Set up a key with a key label with a valid length and try again.

GLD3050A The format of pwEncryption with DES is incorrect in the configuration file.

Severity: Immediate Action

Explanation: The format of `pwEncryption` with DES should be **DES:***keylabel*.

System Action: The program continues. Backends that configure successfully will be available. If no backends configure successfully, an additional message will appear and the program will end.

Administrator Response: Format the `pwEncryption` value correctly and restart the server.

GLD3053E Attention: ldif2tdbm has detected a request for the *requested_step* processing step but the previous step may not have completed successfully.

Severity: Eventual Action

Explanation: This invocation of **ldif2tdbm** uses the same output datasets (specified using the **-o** option) as a previous invocation, hence is seen as a continuation of processing of the earlier invocation. The **ldif2tdbm** program compares the current status in the status file with the processing steps specified on the command to determine if any processing step may have been skipped. Normally, the check (**-c**), prepare (**-p**), and load (**-l**) steps are performed in this order. If the program cannot determine that this order is being followed, this warning is issued. For example, if the current status is L (load) and only **-p** (prepare) is specified on the command, this warning message is issued to indicate that the step before the **-p** step (that is, the check step) may have been skipped. The status for the earlier invocation of **ldif2tdbm** is stored in the `STATUS` record in the status file, *hlq*`.BULKLOAD.JCL(STATUS)`, where *hlq* is the value specified in the **-o** option in the command.

System Action: The program continues to check for other warning conditions, then issues a message prompting whether to continue despite all the warning messages. If the response is **1** (yes), processing continues. If the response is **0** (no), or any character except **1**, processing ends.

Administrator Response: Use the information in the message, the current status in the `STATUS` record in the status file, and the processing steps specified on the command to determine if any necessary steps have been skipped. If not, respond **1** (yes) to the prompt to allow processing to continue. Otherwise respond **0** (no), or any character except **1**, to stop processing and then try the program again, specifying the appropriate processing steps on the command. You can use the **-a** option to provide a response to the prompt as part of the command invocation.

GLD3054E Attention: the specified list of schema files does not match the list of schema files specified in a previous invocation of ldif2tdbm.

Severity: Eventual Action

Explanation: This invocation of **ldif2tdbm** uses the same output datasets (specified using the **-o** option) as a previous invocation, hence is seen as a continuation of processing of the earlier invocation. The **ldif2tdbm** program has detected that the list of schema files specified in the **-s** and the **-v** options of this invocation is different than on the earlier invocation in one of two ways: the lists contain different names or the order of

names in the lists is different. Since LDIF entries that were checked or prepared using the earlier list of schema file names may no longer be valid if processed using the new list of the schema file names, this warning message is issued. The schema file names for the the earlier invocation of **ldif2tdbm** are stored in the SCHEMA records in the status file, *hlq*.BULKLOAD.JCL(STATUS), where *hlq* is the value specified in the **-o** option in the command.

System Action: The program continues to check for other warning conditions, then issues a message prompting whether to continue despite all the warning messages. If the response is **1** (yes), processing continues. If the response is **0** (no), or any character except **1**, processing ends.

Administrator Response: Determine whether processing the LDIF entries using the new list of schema file names is acceptable. If so, respond **1** (yes) to the prompt to allow processing to continue. Otherwise respond **0** (no), or any character except **1**, to stop processing. Then try the program again, either specifying the earlier value for the **-s** and **-v** options, or specifying the new value for the options but changing the requested processing steps to redo the prior steps. You can use the **-a** option to provide a response to the prompt as part of the command invocation.

GLD3055E Attention: the specified list of LDIF files does not match the list of LDIF files specified in a previous invocation of ldif2tdbm.

Severity: Eventual Action

Explanation: This invocation of **ldif2tdbm** uses the same output datasets (specified using the **-o** option) as a previous invocation, hence is seen as a continuation of processing of the earlier invocation. The **ldif2tdbm** program has detected that the list of LDIF input file names specified in the **-i** and the **-e** options of this invocation is different than on the earlier invocation in one of two ways: the lists contain different names or the order of names in the lists is different. Since LDIF entries that were checked or prepared using the earlier list of LDIF input files may no longer be valid for the new list of LDIF input files, this warning message is issued. The LDIF input file names for the earlier invocation of **ldif2tdbm** are stored in the LDIFFILE records in the status file, *hlq*.BULKLOAD.JCL(STATUS), where *hlq* is the value specified in the **-o** option in the command.

System Action: The program continues to check for other warning conditions, then issues a message prompting whether to continue despite all the warning messages. If the response is **1** (yes), processing continues. If the response is **0** (no), or any character except **1**, processing ends.

Administrator Response: Determine whether processing should continue using the new list of LDIF input file names. If so, respond **1** (yes) to the prompt to allow processing to continue. Otherwise respond **0** (no), or any character except **1**, to stop processing. Then try the program again, either specifying the earlier values for the **-i** and **-e** options, or specifying the new values for the options but changing the requested processing steps to redo the prior steps. Note that these options are processed in the order they are specified in the command. You can use the **-a** option to provide a response to the prompt as part of the command invocation.

GLD3056E Attention: ldif2tdbm will overwrite all files that exist in the output datasets during the prepare step.

Severity: Eventual Action

Explanation: During the prepare step, the **ldif2tdbm** program writes files to the output datasets. Since the **ldif2tdbm** program will overwrite the updates from an earlier **ldif2tdbm** program invocation, this warning message is issued. To determine whether to issue this message, the **ldif2tdbm** program reads the STATUS record in the status file, *hlq*.BULKLOAD.JCL(STATUS), where *hlq* is the value specified using the **-o** option. If the STATUS record has a value of P or L, the **ldif2tdbm** program recognizes that an earlier invocation has written to the output datasets and issues this message.

System Action: The program continues to check for other warning conditions, then issues a message prompting whether to continue despite all the warning messages. If the response is **1** (yes), processing continues. If the response is **0** (no), or any character except **1**, processing ends.

Administrator Response: If the output from the previous invocation can be discarded, respond **1** (yes) to the prompt to allow processing to continue. Otherwise respond **0** (no), or any character except **1**, to stop processing and then try the program again, specifying a new valid dataset high level qualifier using the **-o** option. Make sure that all required datasets with the new dataset high level qualifier are allocated before running the program again. You can use the **-a** option to provide a response to the prompt as part of the command invocation.

GLD3057E Attention: attempting to re-run the load step, after ldif2tdbm has successfully completed a load step.

Severity: Eventual Action

Explanation: During the load step (**-l**), the **ldif2tdbm** program submits DB2 load utility jobs and modifies the schema if it is specified (using the **-s** or **-v** options). Once the jobs are submitted, the **ldif2tdbm** program does not have any control over the DB2 load utility jobs. Moreover, resubmitting the DB2 load utility jobs with the same data will result in duplicate data in the database. Hence, the **ldif2tdbm** program should not attempt the

load step again, without preparing new LDIF input files. The **ldif2tdbm** program issues this message when the STATUS record in the status file, *hlq*.BULKLOAD.JCL(STATUS), where *hlq* is the value specified in the **-o** option in the command, has a value of L.

System Action: The program continues to check for other warning conditions, then issues a message prompting whether to continue despite all the warning messages. If the response is **1** (yes), processing continues. If the response is **0** (no), or any character except **1**, processing ends.

Administrator Response: If any of the DB2 load utility jobs submitted by an earlier invocation of the **ldif2tdbm** program completed successfully, respond **0** (no), or any character except **1**, to stop processing. Review the output of each load to determine whether it was successful, refer to the *DB2 Utility Guide and Reference* to correct any problems, and then manually resubmit the DB2 load utility jobs that failed. If all the data prepared by an earlier invocation must be reloaded, respond **1** (yes) to the prompt to allow processing to continue. You can use the **-a** option to provide a response to the prompt as part of the command invocation.

GLD3059E Continue despite previous attention messages? (0=no/1=yes)

Severity: Eventual Action

Explanation: The **ldif2tdbm** program is requesting permission from the administrator to continue despite the warning messages that were previously issued.

System Action: If the administrator responds **1**, the program continues; otherwise, the program ends.

Administrator Response: The administrator must respond to the prompt, answering either **1** (yes) to allow the program to continue or **0** (no), or any character except **1**, to stop processing.

GLD3062I Password encryption method '*method*' is enabled in the TDBM backend.

Severity: Information

Explanation: The LDAP server will encrypt passwords in the TDBM backend using the encryption method specified.

System Action: The program continues.

Administrator Response: None.

GLD3063A An unrecognized status, *value*, encountered in status file.

Severity: Immediate Action

Explanation: The **ldif2tdbm** program has detected an unrecognized value for the STATUS record in the status file. The status file is *hlq*.BULKLOAD.JCL(STATUS), where *hlq* is the value specified in the **-o** option in the command.

System Action: The program ends.

Administrator Response: Correct the value in the STATUS record. Acceptable values are **N** for none, **C** for check, **P** for prepare, or **L** for Load, depending on the last step that **ldif2tdbm** successfully completed.

GLD3064A No value is specified for option: *option*.

Severity: Immediate Action

Explanation: The program was invoked with an option that is missing a value. The option displayed in the message must have a value.

System Action: The program ends.

Administrator Response: Refer to the usage information for the program and try the program again with the proper options and values.

GLD3065A An unrecognized parameter is specified: *parameter*.

Severity: Immediate Action

Explanation: The program was invoked with a parameter that it does not support. The parameter is displayed in the message.

System Action: The program ends.

Administrator Response: Refer to the usage information for the program and try the program again with the proper parameters and values.

GLD3066A Unrecognized value, *value*, for parameter, *parameter*.

Severity: Immediate Action

Explanation: The program was invoked with a value that is not valid for a parameter or option. The parameter and value are displayed in the message.

System Action: The program ends.

Administrator Response: Refer to the usage information for the program and try the program again with the proper parameters and values.

GLD3067A Parameter, *parameter1*, requires parameter, *parameter2*.

Severity: Immediate Action

Explanation: The program was invoked with a parameter that cannot be specified without also specifying a second parameter. The second parameter is missing. The specified parameter and the missing parameter that it requires are displayed in the message.

System Action: The program ends.

Administrator Response: Refer to the usage information for the program and try the program again, making sure that all proper parameter combinations are specified.

GLD3068A A required parameter was not specified.

Severity: Immediate Action

Explanation: The program has been invoked without a required parameter.

System Action: The program ends.

Administrator Response: Refer to the usage information for the program and try the program again, making sure that all required parameters are specified.

GLD3069A A required parameter, *parameter***, was not specified.**

Severity: Immediate Action

Explanation: The program has been invoked without a required parameter. The missing parameter is displayed in the message.

System Action: The program ends.

Administrator Response: Refer to the usage information for the program and try the program again, making sure that all required parameters are specified.

GLD3071A Unrecognized keyword, *keyword***, in status or system file.**

Severity: Immediate Action

Explanation: The **ldif2tdbm** program detected a keyword that is not valid in the status file or the system file. The status file is *hlq*.BULKLOAD.JCL(STATUS) and the system file is *hlq*.BULKLOAD.JCL(SYSTEM), where *hlq* is the value specified in the **-o** option on the **ldif2tdbm** command.

System Action: The program ends.

Administrator Response: Use the information in the record to locate the keyword in the status or system file. Either correct the keyword or remove the record. Then try the program again.

GLD3072A An error occurred while reading line *line_number* **of file** *file_name***.**

Severity: Immediate Action

Explanation: The program encountered an error while trying to read the line specified in the message.

System Action: The program ends.

Administrator Response: Ensure that the file exists and that the specified line can be read. Then try the program again.

GLD3073A A second DN record is found in an LDIF entry at line *line_number* **of LDIF file** *file_name***.**

Severity: Immediate Action

Explanation: The **ldif2tdbm** program encountered an error while processing an entry in an LDIF file. The entry has more than one DN record. A DN record is a record that begins with dn:.

System Action: The program continues check processing if this entry is not the first entry in the first LDIF file and if the check step has been requested; otherwise, the program ends.

Administrator Response: Use the information in the message to locate the entry in the LDIF file and remove one of the DN records. Then try the program again.

GLD3074A A DN record without a value is found in an LDIF entry at line *line_number* **of LDIF file** *file_name***.**

Severity: Immediate Action

Explanation: The **ldif2tdbm** program encountered an error while processing an entry in an LDIF file. The entry has a DN record without any value. A DN value is required for each entry. A DN record is a record that begins with dn:.

System Action: The program continues check processing if this entry is not the first entry in the first LDIF file and if the check step has been requested; otherwise, the program ends.

Administrator Response: Use the information in the message to locate the entry in the LDIF file and add a value to the DN record. Then try the program again.

GLD3075A A non-comment record outside an entry is found at line *line_number* **of LDIF file** *file_name***.**

Severity: Immediate Action

Explanation: The **ldif2tdbm** program encountered an error while processing an LDIF file. The specified record is not part of an entry.

System Action: The program continues check processing if this entry is not the first entry in the first LDIF file and if the check step has been requested; otherwise, the program ends.

Administrator Response: Use the information in the message to locate the record in the LDIF file. Either remove the record or move it so that it is within an entry. Then try the program again.

GLD3076A A value that cannot be processed is found in an LDIF entry at line *line_number* of LDIF file *file_name*.

Severity: Immediate Action

Explanation: The **ldif2tdbm** program encountered an error while processing an entry in an LDIF file. The value that cannot be processed could be the type or value part of the record.

System Action: The program continues check processing if this entry is not the first entry in the first LDIF file and if the check step has been requested; otherwise, the program ends.

Administrator Response: Use the information in the message to locate the record in the LDIF file. Ensure that both the type and value are valid. Then try the program again.

GLD3077A A required change indication line is missing in an LDIF modify entry at line *line_number* of LDIF file *file_name*.

Severity: Immediate Action

Explanation: The **ldif2tdbm** program encountered an error while processing a modify entry in a schema LDIF file. The **ldif2tdbm** program requires that each change clause group in a modify entry begin with a change indication line. There is no default value for this line.

System Action: The program ends because it cannot process the requested changes to the schema.

Administrator Response: Use the information in the message to locate the entry in the LDIF file and add the appropriate change indicator line. Then try the program again.

GLD3078A Incorrect syntax is detected in an entry at line *line_number* of LDIF file *file_name*.

Severity: Immediate Action

Explanation: The **ldif2tdbm** program encountered an error while processing a modify entry in a schema LDIF file. Every line in an entry must follow the required syntax, which is *type*: *value* or *type*:: *value*.

System Action: The program ends because it cannot process the requested changes to the schema.

Administrator Response: Use the information in the message to locate the entry in the LDIF file and correct the syntax of the line. Then try the program again.

GLD3079A A change type that is not supported is found in an LDIF modify entry at line *line_number* of LDIF file *file_name*.

Severity: Immediate Action

Explanation: The **ldif2tdbm** program encountered an error while processing a modify entry in a schema LDIF file. The entry contains a change type line specifying a change type other than **modify**. The schema entry can only be modified.

System Action: The program ends because it cannot process the requested changes to the schema.

Administrator Response: Use the information in the message to locate the entry in the schema LDIF file and remove the entire change record. Then try the program again.

GLD3082A The following option is not supported: *option*.

Severity: Immediate Action

Explanation: The program was invoked with an option that it does not support. The option that is not valid is displayed in the message.

System Action: The program ends.

Administrator Response: Refer to the usage information for the program and try the program again with the proper options.

GLD3083A The DN value specified for the -s option contains characters that are not valid.

Severity: Immediate Action

Explanation: The **tdbm2ldif** program was invoked with a subtree DN value for the **-s** option that contains some characters that could not be converted or normalized.

System Action: The program ends.

Administrator Response: Ensure that the subtree DN value contains only valid characters. Then try the program again.

GLD3084A No TDBM database suffix contains DN *DN*.

Severity: Immediate Action

Explanation: The program cannot find a TDBM database that includes a suffix that can contain the DN displayed in the message. Either no database suffix contains the DN or the database containing the DN is not a TDBM database.

For the **ldif2tdbm** program, this DN is either the DN of the first entry in the first non-empty schema LDIF file (if the **-s** or **-v** option is specified) or the DN of the first

entry of the first non-empty LDIF input file.

For the **tdbm2ldif** program, this is the subtree DN value specified for the **-s** option.

System Action: The program ends.

Administrator Response: Ensure that the configuration file used by the program includes the TDBM database that contains the DN displayed in the message and that this database is correctly configured. Also check that the syntax of the DN value is valid. Then try the program again.

GLD3085A The tdbm2ldif program found more than one TDBM database section. Either use the -s or -n option to specify which TDBM section to process or remove all but one of the TDBM sections from the configuration file.

Severity: Immediate Action

Explanation: When the **tdbm2ldif** program is invoked without specifying the **-s** or the **-n** option, the program fails if there is more than one TDBM database section in the configuration file because it cannot determine which TDBM database to process.

System Action: The program ends.

Administrator Response: The **tdbm2ldif** program provides two options that are used to specify which one of the TDBM database sections in the configuration file to process. These options cannot both be specified at the same time.

- The **-s** option specifies a subtree whose entries are to be unloaded. The **tdbm2ldif** program selects the TDBM database section that contains this subtree from the configuration file.
- The **-n** option indicates the name of a TDBM database section whose entries are to be unloaded. The **tdbm2ldif** program selects the TDBM database section with this name from the configuration file.

Alternatively, you can modify the configuration file and remove all the TDBM database sections except for the one you want to process.

Then try the program again.

GLD3086A Error in internal routine *routine_name***, rc =** *return_code***.**

Severity: Immediate Action

Explanation: An unexpected error occurred in an internal routine used by the program. The name of the routine and its return code are displayed in the message.

System Action: The program ends.

Administrator Response: Use the information in any error messages issued by the routine to fix the problem. Then try the program again. If the problem persists, contact the service representative.

GLD3087A There is no entry in the LDAP directory for DN *DN***.**

Severity: Immediate Action

Explanation: The DN displayed in the message has no entry in the LDAP directory. For the **tdbm2ldif** program, the DN is the subtree DN value specified for the **-s** option.

System Action: The program ends.

Administrator Response: For the **tdbm2ldif** program, either change the subtree DN value to be the DN for an existing entry in the LDAP directory or remove the **-s** option. Then try the program again.

GLD3088A SQL call returned unexpected return code *return_code* **on call:** *SQL_call***.**

Severity: Immediate Action

Explanation: The program cannot continue because an unexpected return code was received from an SQL call. The return code and the SQL call are displayed in the message.

System Action: The program ends.

Administrator Response: Use the information in the message to fix the problem. Then try the program again.

GLD3089A Cannot write record to output file *file_name***. Reason is:** *reason_text***.**

Severity: Immediate Action

Explanation: The program was not able to write a record to an output file. The output file can be a physical file or stdout (standard out). The reason text returned by the system is displayed in the message.

System Action: The program ends.

Administrator Response: Use the information in the message to fix the problem. Ensure that the directory or dataset name displayed in the message exists and that the file can be written. Also check that the file system or dataset is not full. Then try the program again.

GLD3090A The schema LDIF file, *file_name***, contains an entry whose DN is not cn=schema,***suffix* **where** *suffix* **is a suffix for this backend.**

Severity: Immediate Action

Explanation: The **ldif2tdbm** program was invoked with a schema LDIF file that cannot be processed because it contains an entry that is not a schema entry

or that does not include a sufffix that belongs to this TDBM backend.

System Action: The program ends.

Administrator Response: Ensure that all the entries in the schema LDIF file specified in the message are schema entries. A schema entry DN must begin with `cn=schema` and the rest of the DN must be a suffix for this TDBM backend. The TDBM backend in use is the one which contains the suffix included in the DN of the first entry in the first schema file.

GLD3091A An internal exception occurred, rc = *return_code*. Exception text is: *exception_text*.

Severity: Immediate Action

Explanation: An unexpected error occurred in an internal routine used by the program. The return code from the routine and the exception text are displayed in the message.

System Action: The program ends.

Administrator Response: Use the information in the message to resolve the problem. For more detailed error information, issue the command again, specifying the debug error option and use the debug information to try to resolve the problem. If the problem persists, contact the service representative.

GLD3092E Attention: an input file, *file_name*, has been changed since it was last used.

Severity: Eventual Action

Explanation: This invocation of **ldif2tdbm** uses the same output datasets (specified using the **-o** option) as a previous invocation, hence is seen as a continuation of processing of the earlier invocation. The **ldif2tdbm** program has detected that one of the input files specified in the **-s**, **-v**, **-i**, or **-e** option on this invocation has been modified since the earlier invocation. Since LDIF entries that were checked or prepared using the earlier version of this file may no longer be valid for the new contents of this file, this warning message is issued. The time (in seconds since the epoch) of when the earlier invocation of **ldif2tdbm** was run is stored in the `TIME` record in the status file, *hlq*`.BULKLOAD.JCL(STATUS)`, where *hlq* is the value specified in the **-o** option in the command.

System Action: The program continues to check for other warning conditions, then issues a message prompting whether to continue despite all the warning messages. If the response is **1** (yes), processing continues. If the response is **0** (no), or any character other than **1**, processing ends.

Administrator Response: Determine whether processing should continue using the changes to the file specified in the message. If so, respond **1** (yes) to the prompt to allow processing to continue. Otherwise respond **0** (no), or any character except **1**, to stop processing. Then try the program again, changing the requested processing steps to redo the prior steps with the modified input file. You can use the **-a** option to provide a response to the prompt as part of the command invocation.

GLD3093E Attention: additional checking suspended for out-of-date files.

Severity: Eventual Action

Explanation: This invocation of **ldif2tdbm** uses the same output datasets (specified using the **-o** option) as a previous invocation, hence is seen as a continuation of processing of the earlier invocation. When **ldif2tdbm** detects that an LDIF input file has been modified, it issues a warning message. To prevent issuing too many of these messages, the program suspends additional file checking and issues this message when it reaches a pre-set limit to the number of messages.

System Action: The program continues to check for other warning conditions, then issues a message prompting whether to continue despite all the warning messages. If the response is **1** (yes), processing continues. If the response is **0** (no), or any character other than **1**, processing ends.

Administrator Response: The previous warning message indicates the last LDIF input file that was checked. You should check the remaining LDIF input files to ensure that they are not out of date or that any changes to the files are acceptable. If so, respond **1** (yes) to the prompt to allow processing to continue. Otherwise respond **0** (no), or any character except **1**, to stop processing. Then try the program again, changing the requested processing steps to redo the prior steps with the modified input files. You can use the **-a** option to provide a response to the prompt as part of the command invocation.

GLD3096A ldif2tdbm cannot add a child entry under the schema directory node.

Severity: Immediate Action

Explanation: The **ldif2tdbm** program has detected an entry whose parent is the schema entry in the directory. The schema node cannot have any children. The child entry cannot be added to the directory.

System Action: The program continues to process the check step if it has been requested; otherwise, the program ends.

Administrator Response: Use the information in the accompanying message to locate the child entry in an LDIF input file and remove it from the file. Then try the program again.

GLD3097A ldif2tdbm cannot add a child entry without a parent.

Severity: Immediate Action

Explanation: The **ldif2tdbm** program cannot add the LDIF entry to the directory because it cannot locate the parent of the entry. The parent either must already be in the database, or must be an entry before this entry in this LDIF input file or in an LDIF input file processed before this file.

System Action: The program continues to process the check step if it has been requested; otherwise, the program ends.

Administrator Response: Use the information in the accompanying message to locate the child entry in an LDIF input file. Either remove the child entry or add an entry for the parent before the child entry. Then try the program again.

GLD3098A Schema entry found outside a schema input file.

Severity: Immediate Action

Explanation: The **ldif2tdbm** program provides the **-s** and **-v** options for specifying one or more schema input files, which contain entries to modify the schema. The program has found a schema entry in a different LDIF input file. The schema entry cannot be added to the directory. A schema entry DN begins with `cn=schema` and the rest of the DN is a suffix for the TDBM backend.

System Action: The program ends.

Administrator Response: Use the information in the accompanying message to locate the entry in an LDIF input file. Move the entry to a schema input file. Then try the program again. You can also use the **ldapmodify** utility or an LDAP client to process the schema entry.

GLD3099A ldif2tdbm cannot add a child entry under a referral object.

Severity: Immediate Action

Explanation: The **ldif2tdbm** program has detected an entry whose parent is a referral object, so the program cannot add the entry to the directory. The entry must instead be added to the directory that contains the actual entry which is referenced by the referral object.

System Action: The program continues to process the check step if it has been requested; otherwise, the program ends.

Administrator Response: Use the information in the accompanying message to locate the child entry in an LDIF input file and remove the child entry from the file. Then try the program again.

GLD3100A ldif2tdbm cannot add a child entry under this parent because the parent has an incomplete entry in the database.

Severity: Immediate Action

Explanation: The **ldif2tdbm** program has detected an entry whose parent is in the database but the program cannot retrieve all the needed information about the parent from the database. The incomplete information is either the ACL attributes of the parent or the set of ancestors of the parent. The child entry cannot be added to the directory.

System Action: The program continues to process the check step if it has been requested; otherwise, the program ends.

Administrator Response: Use the information in the accompanying message to locate the entry in an LDIF input file. Ensure that the database contains complete ACL attributes and ancestor information for the parent of this entry. Then try the program again.

GLD3101A ldif2tdbm cannot add a duplicate entry.

Severity: Immediate Action

Explanation: The **ldif2tdbm** program has detected an entry which is a duplicate. Another entry with the same DN is either already in the database, or is specified in an entry before this entry in this LDIF input file or in an LDIF input file processed before this file. The duplicate entry cannot be added to the directory.

System Action: The program continues to process the check step if it has been requested; otherwise, the program ends.

Administrator Response: Use the information in the accompanying message to locate the duplicate entry in an LDIF input file. If this entry is a duplicate of a previous entry in an LDIF input file, remove one of these entries. If the entry is a duplicate of an entry in the directory, remove the entry from the LDIF input file. Then try the program again.

GLD3102A The entry has failed schema check.

Severity: Immediate Action

Explanation: The **ldif2tdbm** program has detected an entry whose attributes violate the current schema definition. The current schema is the schema in the directory plus any modifications contained in the schema LDIF files specified with the **-s** or **-v** options on the **ldif2tdbm** command. The entry cannot be added to the directory.

System Action: The program continues to process the check step if it has been requested; otherwise, the program ends.

Administrator Response: Use the information in the

accompanying messages to locate the entry in an LDIF file and determine the problem. If the current schema needs to be modified, create an LDIF file containing the modification and specify the file using the **-s** or **-v** option on the **ldif2tdbm** command. Then try the program again.

GLD3104A Unable to open file *file_name*. Return code from fopen is *errno*; reason is: *reason_text*

Severity: Immediate Action

Explanation: The file named in the message cannot be opened in the required way: read for an input file, write for an output file. The error code and reason text from **fopen** are displayed in the message.

System Action: Usually, the program ends. In some cases, the program can continue limited processing.

Administrator Response: For an input file, ensure that the file exists and can be opened for read. For an output file, check that the directory or dataset containing the file exists and that the file can be written to that directory or dataset. Then try the program again.

GLD3105A A creator DN must be specified for the entries to be loaded.

Severity: Immediate Action

Explanation: The **ldif2tdbm** program requires a DN that the program can use to identify the creator of the entries that will be added to the directory, if those entries do not already include a creator attribute. This DN can be specified on the **-b** option of the **ldif2tdbm** command. If the **-b** option is not specified, the program uses the value of the `adminDN` record in the LDAP configuration file. The same value is also used for the modifier of the entries, if those entries do not already include a modifier attribute.

System Action: The program ends.

Administrator Response: Either add the **-b** option to the **ldif2tdbm** command or add the `adminDN` record to the LDAP configuration file. Then try the program again.

GLD3106A The schema has changed since the load files were prepared.

Severity: Immediate Action

Explanation: The **ldif2tdbm** program has terminated load processing because it has detected that the data in the load files may be out of date. The current schema in the directory was modified after the load files were prepared, thus the data in the load files may not be valid for the modified schema. The **ldif2tdbm** program compares the time of when the load files were prepared (saved in the status file) with the time of when the schema was last modified (saved in the DIR_CACHE table in the database) to determine if the load files may not be usable. The time when the load files were prepared (in seconds since the epoch) is stored in the `TIME` record in the status file, *hlq*`.BULKLOAD.JCL(STATUS)`, where *hlq* is the value specified in the **-o** option in the command.

System Action: The program ends.

Administrator Response: Run **ldif2tdbm** with the **-c** and **-p** options to check and prepare a new set of load files using the current schema. Then invoke the program again to load the data. If you are absolutely sure that the data in the load files is valid for the current schema, you can 'trick' **ldif2tdbm** into loading the data by resetting the value of the `TIME` record in the status file to the present time. Then try the program again.

GLD3107A An internal exception occurred creating cache manager, rc = *return_code*. Exception text is: *exception_text*

Severity: Immediate Action

Explanation: An unexpected error occurred in an internal routine used by the program. The return code from the routine and the exception text are displayed in the message.

System Action: The program continues. The backend which encountered the error ends.

Administrator Response: Use the information in the message to resolve the problem. For more detailed error information, issue the command again, specifying the debug error option and use the debug information to try to resolve the problem. If the problem persists, contact the service representative.

GLD3108A An internal exception occurred creating attribute cache , rc = *return_code*. Exception text is: *exception_text*

Severity: Immediate Action

Explanation: An unexpected error occurred in an internal routine used by the program. The return code from the routine and the exception text are displayed in the message.

System Action: The program continues. The backend which encountered the error ends.

Administrator Response: Use the information in the message to resolve the problem. For more detailed error information, issue the command again, specifying the debug error option and use the debug information to try to resolve the problem. If the problem persists, contact the service representative.

GLD3109A Initialization of database tables failed, rc = *return_code***.**

Severity: Immediate Action

Explanation: An unexpected error occurred in an internal routine while attempting to initialize database tables. The return code from the routine indicates the type of SQL error encountered.

System Action: The program continues. The backend which encountered the error ends.

Administrator Response: Use the information in the message to resolve the problem. For more detailed error information, issue the command again, specifying the debug error option and use the debug information to try to resolve the problem. If the problem persists, contact the service representative.

GLD3110A Error formatting data for *table_name* **tables, rc =** *return_code***.**

Severity: Immediate Action

Explanation: An unexpected error occurred in an internal routine used by the program to format data before updating the database. The return code from the routine is displayed in the message

System Action: The program continues. The backend which encountered the error ends.

Administrator Response: Use the information in the message to resolve the problem. For more detailed error information, issue the command again, specifying the debug error option and use the debug information to try to resolve the problem. If the problem persists, contact the service representative.

GLD3111A SQL Error updating *table_name* **tables, rc =** *return_code***.**

Severity: Immediate Action

Explanation: An unexpected error occurred in an internal routine while attempting to update database tables. The return code from the routine indicates the type of SQL error encountered.

System Action: The program continues. The backend which encountered the error ends.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Use the information in the message to resolve the problem. For more detailed error information, issue the command again, specifying the debug error option and use the debug information to try to resolve the problem. If the problem persists, contact the service representative.

GLD3112A An internal exception occurred creating schema cache, rc = *return_code***. Exception text is:** *exception_text***.**

Severity: Immediate Action

Explanation: An unexpected error occurred in an internal routine used by the program. The return code from the routine and the exception text are displayed in the message.

System Action: The program continues. The backend which encountered the error ends.

Administrator Response: Use the information in the message to resolve the problem. For more detailed error information, issue the command again, specifying the debug error option and use the debug information to try to resolve the problem. If the problem persists, contact the service representative.

GLD3113A An error occurred creating schema cache from database entry cn=schema.

Severity: Immediate Action

Explanation: An error occurred creating an internal cache of the schema entry read from the database.

System Action: The program continues. The backend which encountered the error ends.

Administrator Response: Use the information in the message to resolve the problem. For more detailed error information, issue the command again, specifying the debug error option and use the debug information to try to resolve the problem. If the problem persists, contact the service representative.

GLD3114A An unknown internal exception occurred during *process* **processing.**

Severity: Immediate Action

Explanation: An unexpected error occurred in an internal routine used by the program. The name of the routine is displayed in the message.

System Action: The program continues. The backend which encountered the error ends.

Administrator Response: Use the information in the message to resolve the problem. For more detailed error information, issue the command again, specifying the debug error option and use the debug information to try to resolve the problem. If the problem persists, contact the service representative.

GLD3115A The list file *file_name* **does not contain any LDIF input file names.**

Severity: Immediate Action

Explanation: The **ldif2tdbm** program requires LDIF

input files to be specified on the command line. The **-e** option allows specifying a file that contains a list of LDIF input file names. If the **-i** option is not specified and the file specified with the **-e** option does not contain any LDIF input file names, the **ldif2tdbm** program cannot continue.

System Action: The program ends.

Administrator Response: Specify a new file with the **-e** option that contains valid LDIF input file names, add valid LDIF file names to the file specified in the message, or specify LDIF input files with the **-i** option. Then try the program again.

GLD3116A The file *file_name* is missing a required record: *keyword*.

Severity: Immediate Action

Explanation: The **ldif2tdbm** program requires that one `SSID`, one `HLQ`, and at least one `JOBCARD` record be specified in system file. The record specified in the message is missing from the system file.

System Action: The program ends.

Administrator Response: Add the required record to the system. The try the program again.

GLD3118A Record without value found in file *file_name* on line *line_number*.

Severity: Immediate Action

Explanation: The **ldif2tdbm** program uses a system and status file for required information. The format of the records in these files is: a name, followed by any number of blanks, followed by a value. The only valid names for the system file are `HLQ`, `SSID`, and `JOBCARD`. The only valid names for the status file are `STATUS`, `TIME`, `LDIFFILE`, and `SCHEMA`.

System Action: The program ends.

Administrator Response: Either add a value to the record or remove the record from the file. Then try the program again.

GLD3119I A DB2 deadlock/timeout has been detected and the transaction is being retried internally.

Severity: Information

Explanation: The TDBM or GDBM backend detected a DB2 deadlock/timeout. The transaction is being retried.

System Action: The program continues.

Operator Response: None.

Administrator Response: None.

GLD3120A ldif2tdbm could not submit JCL *file_name*.

Severity: Immediate Action

Explanation: The load step of the **ldif2tdbm** program failed when it attempted to submit the JCL file specified in the message to the JES reader. The reason for the failure is explained in a previous message.

System Action: The program ends.

Administrator Response: Use the information in any error messages issued by the command to fix the problem. **Note:** Do not run the **ldif2tdbm** program again because this can add duplicate data to the database. Instead, use the information in the description of the **ldif2tdbm** command in *z/OS Integrated Security Services LDAP Server Administration and Use Guide* to determine how to proceed.

GLD3121A The *record_name* record in the system file *file_name* is not valid.

Severity: Immediate Action

Explanation: The **ldif2tdbm** program uses the information in the system file to create the JCL used to load the database. The program detected a problem with a record in the status file. The problem is either that the format of the value on the first `JOBCARD` record is not valid or that the value on one of the records is too long. The value of the first `JOBCARD` record must begin with the job name, which consists of 2 slashes directly followed by up to 8 characters (for a total length of up to 10 characters). For usage in the JCL, the maximum length is 71 for a `JOBCARD` record value, 55 for an `SSID` record value, and 36 for an `HLQ` record value. **Note:** DB2 may have different length constraints for some of these values. The record values must meet both the DB2 and **ldif2tdbm** length limitations.

System Action: The program continues to process the check step if it has been requested; otherwise, the program ends.

Administrator Response: Correct any problems with the syntax of the first JOBCARD record value or with the lengths of any of the record values in the system file. Then try the program again.

GLD3122A The __passwd function failed; not loaded from a program controlled library.

Severity: Immediate Action

Explanation: The LDAP server with a TDBM backend needs to be loaded from a program controlled dataset for the **__passwd** function to work.

System Action: The client request fails with an operations error. The program continues.

Operator Response: Ensure that the LDAP server

and the TDBM backend are loaded from a program controlled dataset. Stop and start the server after checking the dataset.

GLD3123A TDBM backend __passwd internal error. Program continues.

Severity: Immediate Action

Explanation: The TDBM backend received an unexpected error from the **__passwd()** function while processing a bind request.

System Action: The client request fails with an operations error. The program continues.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: If the problem persists, contact the service representative.

GLD3124E A Kerberos princSubtree does not exist in the directory. Program continues.

Severity: Eventual Action

Explanation: The princSubtree that exists in the REALM object was not found in the directory. The server will continue to operate but unexpected results may occur.

System Action: The program continues but unexpected results may occur.

Administrator Response: Either the princSubtree value was incorrect or the entry needs to be added to the directory. Correct the error and try again.

GLD3127A Unable to establish initial caches.

Severity: Immediate Action

Explanation: The LDAP server TDBM or GDBM backend encountered an error while trying to establish initial caches.

System Action: The program continues. The backend which encountered the error ends.

Administrator Response: One reason for this error may be lack of memory. Ensure the server has enough memory and restart the server. If the problem persists, contact the service representative.

GLD3129E attrOverflowSize out of bounds; using value of *number*.

Severity: Eventual Action

Explanation: The value specified for `attrOverflowSize` in the configuration file is out of bounds or not numeric and has been replaced by the value specified in the message.

System Action: The program continues.

Administrator Response: If a different `attrOverflowSize` is desired, stop the server, correct the configuration file and restart the server. The value can not be larger than the maximum integer supported on the system.

GLD3130E useNativeAuth value not valid; using default value of 'off'.

Severity: Eventual Action

Explanation: The value specified for `useNativeAuth` in the configuration file is not valid; therefore, the value defaults to **off**.

System Action: The program continues. Native authentication support will not be activated in the server.

Administrator Response: If a different `useNativeAuth` value is desired, stop the server, correct the configuration file and restart the server. Valid values are **on**, **yes**, **off** and **no**.

GLD3131E nativeUpdateAllowed value not valid; using default value of 'no'.

Severity: Eventual Action

Explanation: The value specified for `nativeUpdateAllowed` in the configuration file is not valid; therefore, the value defaults to **no**.

System Action: The program continues. Native password update support will not be activated in the server.

Administrator Response: If a different `nativeUpdateAllowed` value is desired, stop the server, correct the configuration file and restart the server. Valid values are **yes** and **no**.

GLD3132I The useNativeAuth configuration option has been enabled with value '*option_value*'.

Severity: Information

Explanation: Native authentication is enabled for this backend.

System Action: None.

Administrator Response: None.

GLD3133A The version (*dbversion*) of the TDBM or GDBM database is not supported by the current program level.

Severity: Immediate Action

Explanation: The program cannot interact with the TDBM or GDBM database due to a level mismatch. This could be due to running an earlier program version

after the TDBM or GDBM database version has been updated.

System Action: The program continues. The backend which encountered the error ends.

Administrator Response: Ensure that the correct program level is being loaded. Refer to the migration section of *z/OS Integrated Security Services LDAP Server Administration and Use* to determine the proper TDBM or GDBM database version for the program level being used.

GLD3134E No -o parameter supplied, writing to standard output.

Severity: Eventual Action

Explanation: The **tdbm2ldif** program requires a location where it can place the LDIF-formatted output. If the **-o** parameter is not supplied, **tdbm2ldif** will write to standard output.

System Action: The program continues.

Administrator Response: No action may be necessary. If the output from **tdbm2ldif** was intended for a file, run the program again, using the **-o** parameter to indicate where the program should place its output. The value of the option must be the path name of a file or a dataset for **tdbm2ldif**.

GLD3135I Grant/Deny ACL support is enabled below suffixes: *suffix-list*.

Severity: Information

Explanation: Grant/Deny style ACLs along with attribute-level permission settings are supported below the specified suffixes.

System Action: The program continues.

Operator Response: None.

Administrator Response: None.

GLD3136I Grant/Deny ACL support is not enabled below suffixes: *suffix-list*.

Severity: Information

Explanation: Grant/Deny style ACLs along with attribute-level permission settings are not enabled below the specified suffixes. The database version indicates that there may be systems accessing the database that do not support the additional formats.

System Action: The program continues.

Operator Response: None.

Administrator Response: Refer to the migration section of *z/OS Integrated Security Services LDAP Server Administration and Use* for information on migrating the database. The database should only be migrated after all servers accessing the database can support the new database formats.

GLD3137I Using TDBM backend named '*name***'.**

Severity: Information

Explanation: The program is processing the TDBM backend named in the message. The name corresponds to the name specified on the database record marking the beginning of the information for this TDBM backend in the configuration file.

System Action: The program continues.

Operator Response: None.

Administrator Response: You can use the name to locate the database section for this TDBM backend in the configuration file and check that this is the backend that you want to process.

GLD3138I Using TDBM backend containing suffix '*suffix***'.**

Severity: Information

Explanation: The program is processing the TDBM backend containing the suffix displayed in the message. This suffix corresponds to the value of the first suffix record for this TDBM backend in the configuration file.

System Action: The program continues.

Operator Response: None.

Administrator Response: You can use the suffix to locate the database section for this TDBM backend in the configuration file and check that this is the backend that you wanted to process.

GLD3139A There is no TDBM backend with name '*name***'.**

Severity: Immediate Action

Explanation: The program cannot find a TDBM backend with the name displayed in the message. For the **tdbm2ldif** program, this name is the value specified for the **-n** option.

System Action: The program ends.

Administrator Response: Ensure that the configuration file used by the program includes a `database` record that contains the name displayed in the message and specifies **tdbm** for the database type. Also make sure that the backend is correctly configured. Then try the program again.

GLD3140A TDBM or GDBM failed during initialization. One of the following is incorrect: database userid='*dbuserid*'; Table name='*tablename*'; Table column='*tablecolumn*'.

Severity: Immediate Action

Explanation: The program cannot find some global information for the TDBM or GDBM backend. Either the database userid, table name, or table column is wrong.

System Action: The program ends.

Administrator Response: Ensure that the `dbuserid` is correct in the configuration file. If the `dbuserid` is correct, then ensure that there are DB2 tables created for this TDBM or GDBM backend.

GLD3141I End Successful Modify DN operation: newrdn => *new_Relative_Distinguished_Name*; deleteoldrdn => *deleteOldRdn_flag*; dn => *Distinguished_Name*

Severity: Information

Explanation: The LDAP server successfully completed a Modify DN operation. This message displays the new RDN, deleteOldRdn flag and target DN values, respectively, in the operation request. This information is provided to track the changes requested by Modify DN operations, because they may potentially affect many entries in the directory.

System Action: The program continues.

Operator Response: None.

Administrator Response: None.

GLD3142I End Successful Modify DN operation: newSuperior => *new_Superior_Distinguished_Name*; dn => *Distinguished_Name*

Severity: Information

Explanation: The LDAP server successfully completed a Modify DN operation. This message displays the new Superior DN and target DN respectively, in the operation request. This information is provided to track the changes requested by Modify DN operations, because they may potentially affect many entries in the directory.

System Action: The program continues.

Operator Response: None.

Administrator Response: None.

GLD3143I End Successful Modify DN operation: control type => *Control_Type*; control criticality => *Control_Criticality*; control value (hexadecimal) => x'*Control_Value*'; dn => *Distinguished_Name*

Severity: Information

Explanation: The LDAP server successfully completed a Modify DN operation. This message displays the contents of any controls and the target DN submitted in the operation request, including control type, control criticality, and control value (shown in hexadecimal format) for each control. This information is provided to track the changes requested by Modify DN operations, because they may potentially affect many entries in the directory.

System Action: The program continues.

Operator Response: None.

Administrator Response: None.

GLD3144A Idif2tdbm cannot add a child entry under an alias entry.

Severity: Immediate Action

Explanation: The **Idif2tdbm** program has detected an entry whose parent is an alias entry, so the program cannot add the new entry to the directory. An alias entry cannot have any children.

System Action: The program continues to process the check step if it has been requested; otherwise, the program ends.

Operator Response: None.

Administrator Response: Use the information in the accompanying message to locate the child entry in an LDIF input file and remove the child entry from the file. Then try the program again.

GLD3145A Idif2tdbm cannot add an entry that is both an alias entry and a referral entry.

Severity: Immediate Action

Explanation: The **Idif2tdbm** program has detected an entry which is both an alias entry and a referral entry. This combination is not supported, so the program cannot add the new entry to the directory.

System Action: The program continues to process the check step if it has been requested; otherwise, the program ends.

Operator Response: None.

Administrator Response: Use the information in the accompanying message to locate the entry in an LDIF input file. Modify the entry so that it is either an alias

entry or a referral entry, but not both. Then try the program again.

GLD3146I Dynamic, nested, and expanded static group membership determination is in use below suffixes: *suffixes*.

Severity: Information

Explanation: Dynamic, nested, and expanded static group membership determination is now activated and there are entries present in the directory that will take advantage of this expanded group determination. The following objectclasses will now be evaluated as group entries: ibm-staticGroup, ibm-dynamicGroup, ibm-nestedGroup groupOfUniqueNames, groupOfNames, accessRole, accessGroup, and groupOfURLs. Entries containing these objectclasses will now be used during group gathering time to see if an authenticating user belongs to any of these groups. With this capability, it is possible to query expanded static, dynamic, and nested groups with the use of the ibm-allGroups and ibm-allMembers operational attributes.

System Action: The program continues.

Operator Response: None.

Administrator Response: None.

GLD3147I Dynamic, nested, and expanded static group membership determination is available but not in use below suffixes: *suffixes*.

Severity: Information

Explanation: Dynamic, nested, and expanded static group membership determination is activated, however, there are currently no entries present in the directory that will take advantage of this expanded group determination. With this expanded capability, the following objectclasses will now be evaluated as group entries: ibm-staticGroup, ibm-dynamicGroup, ibm-nestedGroup, groupOfUniqueNames, groupOfNames, accessRole, accessGroup, and groupOfURLs. Entries containing these objectclasses will now be used during group gathering time to see if an authenticating user belongs to any of these groups. With this capability, it is possible to query expanded static, dynamic, and nested groups with the use of the ibm-allGroups and ibm-allMembers operational attributes.

System Action: The program continues.

Operator Response: None.

Administrator Response: Large static group entries can now be migrated to either nested or dynamic groups. Please refer to *z/OS Integrated Security Services LDAP Server Administration and Use* for instructions on how to form dynamic and/or nested groups to simplify management of large static groups.

GLD3148I Dynamic, nested, or expanded static group data is present in the TDBM backend but ignored since the DB_VERSION is not 3.0 or greater below suffixes: *suffixes*.

Severity: Information

Explanation: Dynamic, nested, or expanded static groups are present in the backend, however, are ignored since the TDBM DB_VERSION has not been updated to at least 3.0. The only groups that are currently being evaluated during group gathering in the TDBM backend are static groups that have an objectclass of accessGroup. These are also the only groups that are currently used in ibm-allMembers and ibm-allGroups search and comparison operations.

System Action: The program continues.

Operator Response: None.

Administrator Response: In order to support dynamic, nested, and expanded static group membership, the DB_VERSION should be set to at least 3.0. Please refer to *z/OS Integrated Security Services LDAP Server Administration and Use* for information on migrating the database.

GLD3149I DN: *group_dn* **has an invalid memberURL attribute value of** *memberurl_val*.

Severity: Information

Explanation: The memberURL attribute specified in this entry will not be evaluated for dynamic group membership determination and will be ignored for dynamic group querying and searching.

System Action: The program continues.

Operator Response: None.

Administrator Response: Update the entry's memberURL attribute to the correct format so that it can be properly evaluated for dynamic group membership.

GLD3150E The *option* **option is not supported in multi-server mode. The option is ignored.**

Severity: Eventual Action

Explanation: The LDAP server configuration option displayed in the message was specified, but the database for this backend is running in multi-server mode. This option is not supported in multi-server mode. The option is ignored.

System Action: The program continues. The option is ignored.

Administrator Response: Correct the configuration file and restart the server.

GLD3151I The backend containing the following suffix is participating in change logging: '*suffix*'.

Severity: Information

Explanation: This backend has requested to participate in change logging. This is a result of either not specifying the `changeLoggingParticipant` option in the section for this backend in the configuration file or specifying a value of `yes` for this configuration option. Change logging must be configured and turned on for change log entries to be created. For a TDBM backend, any change to an entry (including the schema) in this backend will result in the creation of a change log entry for the change. For a GDBM backend, only changes to the change log root entry or to the schema are logged.

System Action: None.

Administrator Response: None. If you do not want to create change log entries for changes to entries in this backend, then specify `changeLoggingParticipant no` in the section for this backend in the configuration file. Then restart the server.

GLD3152I The backend containing the following suffix is not participating in change logging: '*suffix*'.

Severity: Information

Explanation: This backend has chosen to not participate in change logging. A change to an entry in this backend will not result in the creation of a change log entry for the change. This is a result of specifying the `changeLoggingParticipant no` option in the section for this backend in the configuration file.

System Action: None.

Administrator Response: None. If you do want to create change log entries for changes to entries in this backend, then either specify `changeLoggingParticipant yes` or remove the `changeLoggingParticipant` option from the section for this backend in the configuration file. Also check that change logging is enabled and turned on in the configuration file. Then restart the server.

GLD3153A A GDBM and TDBM backend cannot have the same dbuserid value, *database_owner*.

Severity: Immediate Action

Explanation: The configuration file contains a GDBM section and a TDBM section that have the same `dbuserid` value. GDBM and TDBM cannot share a DB2 database, thus must specify a different value for the database owner.

System Action: The program ends.

Administrator Response: Change the `dbuserid` value for either the GDBM section or the TDBM section. This value must match the owner of the tables in the SPUFI scripts used to create the database. Then start the server again.

LDAP server product messages (4000)

GLD4001A Requested message number *max_number* beyond bounds of internal table.

Severity: Immediate Action

Explanation: The message specified by the message number cannot be displayed because it is not in the internal message table and was not found in any catalog.

System Action: The program continues.

Operator Response: Contact the service representative. An internal error has occurred.

GLD4002E Cannot open message catalog file *file_name*.

Severity: Eventual Action

Explanation: The program was unable to locate the specified message catalog. Possible reasons include: message catalogs installed incorrectly or, **LANG** or **NLSPATH** environment variables may not be set or may be set incorrectly.

System Action: The program continues.

Operator Response: Verify the full path name to the message catalogs. Verify that the **LANG** and **NLSPATH** variables have the correct values to reach this path. Ensure the message catalogs are installed correctly and have the correct permissions. If the problem persists, contact your service representative.

GLD4003A Memory allocation failed; cannot continue. Program terminated.

Severity: Immediate Action

Explanation: An LDAP facility was unable to allocate the necessary memory to continue processing. The program is ending.

System Action: The program ends.

Operator Response: Ensure that the program has sufficient memory and try again. If the problem persists, contact the service representative.

GLD4004A Unable to write error message to console. Return code=*return_code*.

Severity: Immediate Action

Explanation: An LDAP program received an error from system routine **__console()** when attempting to write a message to the operator's console.

System Action: The program continues.

Operator Response: Contact the service representative.

GLD4005E Environment variable file not found. Environment variables not set. Continuing.

Severity: Eventual Action

Explanation: The LDAP program was unable to find the specified environment variable file. Program continues with environment variables not set.

System Action: The program continues.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Verify the existence and location of the environment variable file and restart the server.

GLD4006E Environment variable file cannot be opened. Environment variables not set. Continuing.

Severity: Eventual Action

Explanation: The LDAP program was unable to open the environment variable file. Program continues with environment variables not set.

System Action: The program continues.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Verify that the file system is operating correctly and restart the server.

GLD4007E Environment variable file contents in error. Environment variables not set. Continuing.

Severity: Eventual Action

Explanation: The LDAP program encountered an incorrect line in its environment variable file. Program continues with environment variables not set.

System Action: The program continues.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Verify the contents of the environment variable file and restart the server.

GLD4008E Environment variables not set because error encountered. Continuing.

Severity: Eventual Action

Explanation: The LDAP program encountered an unexpected error when processing its environment variable file. Program continues with environment variables not set.

System Action: The program continues.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: For more detailed error information, issue the command again, specifying the debug error option and use the debug information to try to resolve the problem. If the problem persists, contact the service representative.

GLD4009A The initialization of ServerGlobal structure failed.

Severity: Immediate Action

Explanation: The LDAP program encountered an error when establishing the ServerGlobal data structure. The program cannot continue without this structure.

System Action: The program ends.

Administrator Response: Ensure adequate storage for the program and try again. If the problem persists, contact the service representative.

GLD4010A The initialization of ConfigInfo structure failed.

Severity: Immediate Action

Explanation: The LDAP program encountered an error when establishing the ConfigInfo data structure. The program cannot continue without this structure.

System Action: The program ends.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Ensure adequate storage for the program and try again. If the problem persists, contact the service representative.

GLD4012A The initialization of ReplInfo structure failed.

Severity: Immediate Action

Explanation: The LDAP program encountered an error when establishing the ReplInfo data structure. The program cannot continue without this structure.

System Action: The program ends.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Ensure adequate storage for the program and try again. If the problem persists, contact the service representative.

GLD4013A Unable to locate specified catalog.

Severity: Immediate Action

Explanation: The LDAP program encountered an error when attempting to issue a message. The program will continue but may not issue appropriate messages.

System Action: The program continues.

Operator Response: Contact the service representative.

GLD4014A Unable to establish message support.

Severity: Immediate Action

Explanation: The LDAP program encountered an error when attempting to establish message support.

System Action: The program ends.

Operator Response: Contact the service representative.

GLD4015A An error occurred when reading the initial schema.

Severity: Immediate Action

Explanation: The LDAP server or a server backend encountered an error while trying to read the initial or minimum schema.

System Action: If the problem occurs while processing a backend, the server continues to process any other backends. Otherwise, the server ends.

Administrator Response: If the initial schema is being read from a file, make sure that the file exists and can be read. Additional information is provided using debug messages when the debug level is set to LDAP_DEBUG_ERROR. If the problem persists, contact the service representative.

GLD4016A Unable to establish initial schema. Return code is: *return_code***; reason is:** *reason_text*

Severity: Immediate Action

Explanation: The LDAP server or a server backend encountered an error while trying to establish the initial or minimum schema.

System Action: If the problem occurs while processing a backend, the server continues to process any other backends. Otherwise, the server ends.

Administrator Response: Use the LDAP return code and reason text displayed in the message to resolve the problem. Then start the server again. If the problem persists, contact the service representative.

GLD4017A Error in internal routine *routine_name***, rc =** *return_code***.**

Severity: Immediate Action

Explanation: An unexpected error occurred in an internal routine used by the program. The name of the routine and its return code are displayed in the message.

System Action: If the program is the LDAP server and the problem occurs while processing a backend, the server continues to process any other backends. Otherwise, the program ends.

Administrator Response: Use the information in any error or debug messages issued by the routine to fix the problem. Then try the program again. If the problem persists, contact the service representative.

GLD4018A Option '*option_name***' is specified more than once with different values.**

Severity: Immediate Action

Explanation: The invocation of this program includes an option that is specified more than once with a different value. The program cannot determine which value to use for the option. The option is displayed in the message.

System Action: If the program is the LDAP server and the problem occurs while processing a backend, the server continues to process any other backends. Otherwise, the program ends.

Administrator Response: Ensure that the option is specified only once and has the desired value. Then try the program again.

GLD4019A Do not specify both of the following options: '*option1***' and '***option2***'.**

Severity: Immediate Action

Explanation: The program was invoked with two options that are mutually exclusive. You cannot specify both of these options.

System Action: The program ends.

Administrator Response: Refer to the usage information for the program and try the program again with the proper options.

GLD4020I The following LDAP message is truncated at the console.

Severity: Information

Explanation: The next console message from LDAP (prefix GLD) is longer than the console message limit.

See the LDAP server log to view the complete message.

System Action: The program continues.

Operator Response: None.

Administrator Response: See the server log for the complete message.

GLD4021E Environment variable file line too long, line *linenum* **file** *filename***, continuing.**

Severity: Eventual action

Explanation: The LDAP program encountered an incorrect line in its environment variable file. Program continues with environment variable on the specified line not set.

System Action: The program continues.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Verify the contents of the environment variable file and restart the server.

GLD4022E Environment variable missing '=', line *linenum* **file** *filename***, continuing.**

Severity: Eventual action

Explanation: The LDAP program encountered an incorrect line in its environment variable file. Program continues with environment variable on the specified line not set.

System Action: The program continues.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Verify the contents of the environment variable file and restart the server.

GLD4023E Environment variable NULL, line *linenum* **file** *filename***, continuing.**

Severity: Eventual Action

Explanation: The LDAP program encountered an incorrect line in its environment variable file. Program continues with environment variable on the specified line not set.

System Action: The program continues.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Verify the contents of the environment variable file and restart the server.

GLD4024E Environment variable could not be set, line *linenum* **file** *filename***, continuing.**

Severity: Eventual Action

Explanation: An attempt to set an environment variable failed. Program continues with environment variable on the specified line not set.

System Action: The program continues.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Verify the contents of the environment variable file and restart the server.

GLD4025E Last line in environment variable file has continuation character, line *linenum* **file** *filename***, continuing.**

Severity: Eventual Action

Explanation: The LDAP program encountered an incorrect line in its environment variable file. Program continues with environment variable on the specified line not set.

System Action: The program continues.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Verify the contents of the environment variable file and restart the server.

SDBM messages (5000)

GLD5001A Memory allocation failed; cannot continue. Program terminated.

Severity: Immediate Action

Explanation: The SDBM backend was unable to allocate the necessary memory to continue processing. The program is ending.

System Action: The program ends.

Operator Response: Ensure that the LDAP server has sufficient memory and try again. If the problem persists, contact the service representative.

GLD5002A The __passwd() function failed; not loaded from a program controlled library.

Severity: Immediate Action

Explanation: The LDAP server with an SDBM backend needs to be loaded from a program controlled dataset for the **__passwd()** function to work.

System Action: The client request fails with an operations error. The program continues.

Operator Response: Ensure that the LDAP server and the SDBM backend are loaded from a program controlled dataset. Stop and start the server after checking the dataset.

GLD5003A SDBM backend __passwd() internal error. Program continues.

Severity: Immediate Action

Explanation: The SDBM backend received an unexpected error from the **__passwd()** function while processing a bind request.

System Action: The client request fails with an operations error. The program continues.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: If the problem persists, contact the service representative.

GLD5004A *keyword* **parameter is missing or has no value in the configuration file.**

Severity: Immediate Action

Explanation: The SDBM backend for the LDAP server cannot be configured because the configuration file is missing a required parameter or the parameter has no value.

System Action: The program continues. Backends that configure successfully will be available. If no backends configure successfully, an additional message will appear and the program will end.

Administrator Response: Correct the configuration file and try again.

GLD5005A RDN in suffix '*suffix***' is not valid.**

Severity: Immediate Action

Explanation: The value for the suffix parameter is not valid in the SDBM backend section of the configuration file. The specified Relative Distinguished Name (RDN) is not correct.

System Action: The program continues. Backends that configure successfully will be available. If no backends configure successfully, an additional message will appear and the program will end.

Administrator Response: Correct the suffix value in the configuration file. Make sure that the RDN consists of an attribute type and value, separated by an equal sign.

GLD5006A Only one suffix can be specified for the SDBM backend in the configuration file.

Severity: Immediate Action

Explanation: The SDBM backend for the LDAP server cannot be configured because the configuration file contains multiple suffix lines for SDBM. There can only be one one SDBM backend section in the configuration file and it must contain only one suffix line.

System Action: The program continues. Backends that configure successfully will be available. If no backends configure successfully, an additional message will appear and the program will end.

Administrator Response: Correct the configuration file and try again.

ldapcnf messages (7000)

GLD7001I Usage: ldapcnf -i profile_file

Severity: Information

Explanation: The correct syntax of the **ldapcnf** command is: **ldapcnf -i** *profile_file*

System Action: The program ends.

Operator Response: Invoke the **ldapcnf** command again with the appropriate options.

Administrator Response: None.

GLD7002I Generating *output_file* ....

Severity: Information

Explanation: Status message which indicates the generation of an output file by the **ldapcnf** utility.

System Action: The program continues.

Administrator Response: None.

GLD7003I Finished generating *output_file*.

Severity: Information

Explanation: Status message which indicates the completed generation of an output file by the **ldapcnf** utility.

System Action: The program continues.

Administrator Response: None.

GLD7004A Errors found. Correct errors and try again.

Severity: Immediate Action

Explanation: The **ldapcnf** utility detected errors.

System Action: The program ends.

Administrator Response: Use the information in any error messages issued by the utility to fix the problem. Then try the command again.

GLD7005I The utility is finished checking for errors.

Severity: Information

Explanation: Status message which indicates that **ldapcnf** has completed error checking.

System Action: The program continues.

Administrator Response: None.

GLD7006A *statement_name* is missing in the input file *input_file*.

Severity: Immediate Action

Explanation: The **ldapcnf** utility has detected a missing statement in the input file.

System Action: The program continues.

Administrator Response: Correct the missing statement in the input file.

GLD7007A Value *statement_name* is greater than *length* characters in the input file *input_file*.

Severity: Immediate Action

Explanation: The **ldapcnf** utility has detected that the statement value is greater than the specified limit in the input file.

System Action: The program continues.

Administrator Response: If the value should be greater than the limit specified, see the Usage Notes for the **ldapcnf** utility. Otherwise correct the value in the input file.

GLD7008A Value for *statement_name* contains nonprintable characters in the input file *input_file*.

Severity: Immediate Action

Explanation: The **ldapcnf** utility has detected that a specified value in the input file contains nonprintable characters.

System Action: The program continues.

Administrator Response: Correct the statement value in the input file.

GLD7009E Attention: The output data set *data_set_name* has been previously used. Do you wish to overwrite the existing members of this data set? (y/n) [n]

Severity: Eventual Action

Explanation: The output data set specified on **ldapcnf** invocation already contains outputs from the command. This is prompting the user to specify if they wish to overwrite existing members in the output data set. If the output data set is currently being used for an LDAP server, a different output data set should be used for this invocation of the **ldapcnf** utility.

System Action: The program continues or ends based upon input.

Administrator Response: Specify **y** or **n**.

GLD7010A The temporary directory (TEMPORARY_DIR statement in the input file *input_file*) specified (*directory_name*) does not exist.

Severity: Immediate Action

Explanation: The **ldapcnf** utility has detected that the directory specified on the TEMPORARY_DIR statement does not exist.

System Action: The program ends.

Administrator Response: Correct the temporary directory statement (TEMPORARY_DIR) value in the input file.

GLD7011I Exiting with return code *return_code*.

Severity: Information

Explanation: The **ldapcnf** utility is exiting.

System Action: The program ends.

Administrator Response: If the return code is nonzero, look for previous error messages.

GLD7012I Terminating upon user request.

Severity: Information

Explanation: The **ldapcnf** utility has been terminated by the user.

System Action: The program ends.

Administrator Response: None.

GLD7013A Value for *statement_name1* must be different from the value for *statement_name2*.

Severity: Immediate Action

Explanation: The **ldapcnf** utility has detected two statement values that are equal.

System Action: The program ends.

Administrator Response: Correct one of the statement values, making sure that the two statement values are different.

Extended operations messages (9000)

GLD9001A An error occurred during *backend_type* backend initialization, rc = *return_code*.

Severity: Immediate Action

Explanation: An error occurred during backend initialization.

System Action: The program continues. Backends that configure successfully will be available. If no backends configure successfully, an additional message will appear and the program will end.

Administrator Response: Use the information in the message to resolve the problem. For more detailed error information, issue the command again, specifying the debug error option and use the debug information to try to resolve the problem. If the problem persists, contact the service representative.

GLD9002A An unknown internal exception occurred during an extended operation.

Severity: Immediate Action

Explanation: An unexpected error occurred in an internal routine used by the program.

System Action: The program continues.

Administrator Response: Contact the service representative.

GLD9003A An internal exception occurred during an extended operation. Return code is *errno*; error string is: *error_string*.

Severity: Immediate Action

Explanation: An unexpected error occurred in an internal routine used by the program.

System Action: The program continues. The request fails.

Administrator Response: Use the information in the message to resolve the problem. For more detailed error information, issue the command again, specifying the debug error option and use the debug information to try to resolve the problem. If the problem persists, contact the service representative.

GLD9004E The extended operations backend encountered an error while attempting LDAP client requests. Return code is *ldap_errno*; error string is: *ldap_errstring*.

Severity: Eventual Action

Explanation: To satisfy some extended operations, the extended operations backend uses the LDAP client APIs. While handling such a request, the extended operations backend encountered an error.

System Action: The program continues. The request fails.

Administrator Response: Use the information in the message to resolve the problem. For more detailed error information, issue the command again, specifying the debug error option and use the debug information to try to resolve the problem. If the problem persists, contact the service representative.

GLD9005E The extended operations backend encountered an unrecognizable control. Return code is *ldap_errno*; error string is: *error_string*.

Severity: Eventual Action

Explanation: During an extended operations request, the extended operations backend encountered an unrecognizable control.

System Action: The program continues. The request fails.

Administrator Response: Use the information in the message to resolve the problem. For more detailed error information, issue the command again, specifying the debug error option and use the debug information to try to resolve the problem. If the problem persists, contact the service representative.

GLD9006E The LDAP program encountered an unrecognizable extended operation request. Return code is *ldap_errno***; error string is:** *error_string***.**

Severity: Eventual Action

Explanation: The LDAP program encountered an unrecognizable extended operations request.

System Action: The program continues. The request fails.

Administrator Response: Use the information in the message to resolve the problem. For more detailed error information, issue the command again, specifying the debug error option and use the debug information to try to resolve the problem. If the problem persists, contact the service representative.

GLD9007E The extended operation requires a critical control. Error string is: *error_string***.**

Severity: Eventual Action

Explanation: The LDAP program did not receive a required critical control with an extended operations request.

System Action: The program continues. The request fails.

Administrator Response: Use the information in the message to resolve the problem. For more detailed error information, issue the command again, specifying the debug error option and use the debug information to try to resolve the problem. If the problem persists, contact the service representative.

GLD9008A The extended operation backend could not allocate memory. Error string is: *error_string***.**

Severity: Immediate Action

Explanation: The LDAP program could not allocate memory during its processing.

System Action: The program continues. The request fails.

Operator Response: Contact the LDAP Administrator or see the Administrator Response.

Administrator Response: Ensure adequate storage for the program and try again. If the problem persists, contact the service representative.

Part 4. Appendixes

Appendix A. Minimum schema for TDBM and GDBM

This appendix shows the minimum schema for TDBM assuming the suffix is `o=your company`. This is also the minimum schema for GDBM, except that the suffix is `cn=changelog`.

```
cn=schema,o=your company
cn=SCHEMA
subtreespecification=NULL
objectclass=TOP
objectclass=SUBSCHEMA
objectclass=SUBENTRY
objectclass=IBMSUBSCHEMA
attributetypes=( 1.3.18.0.2.4.285 NAME 'aclentry' EQUALITY caseexactmatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32700} USAGE directoryoperation )
attributetypes=( 1.3.18.0.2.4.286 NAME 'aclpropagate'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7{5} USAGE directoryoperation SINGLE-VALUE )
attributetypes=( 1.3.18.0.2.4.287 NAME 'aclsource'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12{1000} USAGE directoryoperation SINGLE-VALUE NO-USER-MODIFICATION )
attributetypes=( 2.5.4.1 NAME ( 'aliasedObjectName' 'aliasedEntryName' ) EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.6.1.4.1.1466.101.120.6 NAME 'altserver'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE dsaoperation )
attributetypes=( 2.5.21.5 NAME 'attributetypes' EQUALITY objectidentifierfirstcomponentmatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.3 USAGE directoryoperation )
attributetypes=( 2.16.840.1.113730.3.1.5 NAME 'changenumber'
  DESC 'Contains the change number of the entry as assigned by the supplier server.' EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE NO-USER-MODIFICATION USAGE userApplications )
attributetypes=( 2.16.840.1.113730.3.1.8 NAME 'changes'
  DESC 'Defines changes made to a directory server. These changes are in LDIF format.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 SINGLE-VALUE NO-USER-MODIFICATION USAGE userApplications )
attributetypes=( 2.16.840.1.113730.3.1.77 NAME 'changetime' DESC 'Time last changed.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE NO-USER-MODIFICATION USAGE userApplications )
attributetypes=( 2.16.840.1.113730.3.1.7 NAME 'changetype'
  DESC 'Describes the type of change performed on an entry. Accepted values include: add, delete, modify, modrdn.'
  EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE NO-USER-MODIFICATION USAGE userApplications )
attributetypes=( 2.5.4.3 NAME ( 'cn' 'commonname' ) SUP name )
attributetypes=( 2.5.18.1 NAME 'createtimestamp' EQUALITY generalizedtimematch ORDERING generalizedtimeorderingmatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE NO-USER-MODIFICATION USAGE directoryoperation )
attributetypes=( 2.5.18.3 NAME 'creatorsname' EQUALITY distinguishednamematch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE NO-USER-MODIFICATION USAGE directoryoperation )
attributetypes=( 2.16.840.1.113730.3.1.10 NAME 'deleteoldrdn'
  DESC 'a flag which indicates if the old RDN should be retained as an attribute of the entry'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 SINGLE-VALUE NO-USER-MODIFICATION USAGE userApplications )
attributetypes=( 2.5.21.2 NAME 'ditcontentrules' EQUALITY objectidentifierfirstcomponentmatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.16 USAGE directoryoperation )
attributetypes=( 2.5.21.1 NAME 'ditstructurerules' EQUALITY integerfirstcomponentmatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.17 USAGE directoryoperation )
attributetypes=( 2.5.4.49 NAME ( 'dn' 'distinguishedname' ) EQUALITY distinguishednamematch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
attributetypes=( 1.3.18.0.2.4.288 NAME 'entryowner' EQUALITY caseexactmatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{1000} USAGE directoryoperation )
attributetypes=( 1.3.18.0.2.4.470 NAME 'ibmattributetypes' EQUALITY objectidentifierfirstcomponentmatch
  SYNTAX 1.3.18.0.2.8.1 USAGE directoryoperation )
attributetypes=( 1.3.18.0.2.4.2244 NAME 'ibm-allgroups'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 NO-USER-MODIFICATION USAGE directoryoperation )
attributetypes=( 1.3.18.0.2.4.2243 NAME 'ibm-allmembers'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 NO-USER-MODIFICATION USAGE directoryoperation )
attributetypes=( 1.3.18.0.2.4.1155 NAME 'ibm-changeinitiatorsname'
  DESC 'the dn of the entity that initiated the change'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2482 NAME 'ibm-enabledCapabilities' EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{100} USAGE dsaoperation NO-USER-MODIFICATION DESC
  'Lists capabilities enabled for use on this server.'  )
attributetypes=( 1.3.18.0.2.4.1780 NAME 'ibm-entryuuid' EQUALITY ibm-entryuuidmatch
  SYNTAX 1.3.18.0.2.8.3 USAGE dsaoperation NO-USER-MODIFICATION SINGLE-VALUE
  DESC 'Uniquely identifies an ldap entry throughout its life.' )
attributetypes=( 1.3.18.0.2.4.2242 NAME 'ibm-membergroup' EQUALITY distinguishednamematch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userapplications )
attributetypes=( 1.3.18.0.2.4.2481 NAME 'ibm-supportedCapabilities' EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{100} USAGE dsaoperation NO-USER-MODIFICATION
  DESC 'Lists capabilities supported, but not necessarily enabled, by this server.'  )
attributetypes=( 0.9.2342.19200300.100.1.24 NAME 'lastmodifiedby'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12{1000} SINGLE-VALUE )
attributetypes=( 0.9.2342.19200300.100.1.23 NAME 'lastmodifiedtime'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24{30} SINGLE-VALUE )
attributetypes=( 1.3.6.1.4.1.1466.101.120.16 NAME 'ldapsyntaxes' EQUALITY objectidentifierfirstcomponentmatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.54 USAGE directoryoperation )
attributetypes=( 2.5.21.4 NAME 'matchingrules' EQUALITY objectidentifierfirstcomponentmatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.30 USAGE directoryoperation )
attributetypes=( 2.5.21.8 NAME 'matchingruleuse' EQUALITY objectidentifierfirstcomponentmatch
```

```
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.31 USAGE directoryoperation )
  attributetypes=( 2.5.4.31 NAME 'member' SUP distinguishedname )
  attributetypes=( 2.16.840.1.113730.3.1.198 NAME 'memberurl' EQUALITY caseexactmatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userapplications )
  attributetypes=( 2.5.18.4 NAME 'modifiersname' EQUALITY distinguishednamematch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE NO-USER-MODIFICATION USAGE directoryoperation )
  attributetypes=( 2.5.18.2 NAME 'modifytimestamp' EQUALITY generalizedtimematch ORDERING generalizedtimeorderingmatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE NO-USER-MODIFICATION USAGE directoryoperation )
  attributetypes=( 2.5.4.41 NAME 'name' EQUALITY caseignorematch SUBSTR caseignoresubstringsmatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )
  attributetypes=( 2.5.21.7 NAME 'nameforms' EQUALITY objectidentifierfirstcomponentmatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.35 USAGE directoryoperation )
  attributetypes=( 1.3.6.1.4.1.1466.101.120.5 NAME 'namingcontexts'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE dsaoperation )
  attributetypes=( 2.16.840.1.113730.3.1.9 NAME 'newrdn'
    DESC 'the new RDN of an entry which is the target of a modrdn operation.' EQUALITY distinguishedNameMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE NO-USER-MODIFICATION USAGE userApplications )
  attributetypes=( 2.16.840.1.113730.3.1.11 NAME 'newsuperior'
    DESC 'Specifies the name of the entry that will become the immediate superior of the existing entry,
    when processing a modDN operation.' EQUALITY distinguishedNameMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE NO-USER-MODIFICATION USAGE userApplications )
  attributetypes=( 2.5.4.0 NAME 'objectclass' EQUALITY objectidentifiermatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )
  attributetypes=( 2.5.21.6 NAME 'objectclasses' EQUALITY objectidentifierfirstcomponentmatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.37 USAGE directoryoperation )
  attributetypes=( 1.3.18.0.2.4.289 NAME 'ownerpropagate'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.7{5} USAGE directoryoperation SINGLE-VALUE )
  attributetypes=( 1.3.18.0.2.4.290 NAME 'ownersource'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.12{1000} USAGE directoryoperation  SINGLE-VALUE NO-USER-MODIFICATION )
  attributetypes=( 2.16.840.1.113730.3.1.34 NAME 'ref'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{100} EQUALITY caseexactmatch )
  attributetypes=( 1.3.18.0.2.4.299 NAME 'replicabinddn'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.12{1000} USAGE directoryoperation )
  attributetypes=( 1.3.18.0.2.4.302 NAME 'replicabindmethod'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{100} USAGE directoryoperation )
  attributetypes=( 1.3.18.0.2.4.300 NAME ( 'replicacredentials' 'replicabindcredentials' )
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.5{128} SINGLE-VALUE USAGE directoryoperation )
  attributetypes=( 1.3.18.0.2.4.298 NAME 'replicahost'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{100} SINGLE-VALUE USAGE directoryoperation )
  attributetypes=( 1.3.18.0.2.4.301 NAME 'replicaport'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{10} SINGLE-VALUE USAGE directoryoperation )
  attributetypes=( 1.3.18.0.2.4.304 NAME 'replicaupdatetimeinterval'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{20} SINGLE-VALUE USAGE directoryoperation )
  attributetypes=( 1.3.18.0.2.4.303 NAME 'replicausessl'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{10} SINGLE-VALUE USAGE directoryoperation )
  attributetypes=( 2.5.18.10 NAME 'subschemasubentry' EQUALITY distinguishednamematch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE NO-USER-MODIFICATION USAGE directoryoperation )
  attributetypes=( 2.5.18.6 NAME 'subtreespecification'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE USAGE directoryOperation )
  attributetypes=( 1.3.6.1.4.1.1466.101.120.13 NAME 'supportedcontrol'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 USAGE dsaoperation )
  attributetypes=( 1.3.6.1.4.1.1466.101.120.7 NAME 'supportedextension'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 USAGE dsaoperation )
  attributetypes=( 1.3.6.1.4.1.1466.101.120.15 NAME 'supportedldapversion'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE dsaoperation )
  attributetypes=( 1.3.6.1.4.1.1466.101.120.14 NAME 'supportedsaslmechanisms'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE dsaoperation )
  attributetypes=( 2.16.840.1.113730.3.1.6 NAME 'targetdn'
    DESC 'Defines the distinguished name of an entry that was added, modified, or deleted on a supplier server.  In
    the case of a modrdn operation, the targetDn contains the distinguished name of the entry before it was modified.'
    EQUALITY distinguishedNameMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE NO-USER-MODIFICATION USAGE userApplications )
  attributetypes=( 2.5.4.50 NAME 'uniquemember' EQUALITY distinguishednamematch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userapplications )
  attributetypes=( 2.5.4.35 NAME 'userpassword' EQUALITY octetstringmatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.40{128} )
  ibmattributetypes=( 1.3.18.0.2.4.285 ACCESS-CLASS restricted )
  ibmattributetypes=( 1.3.18.0.2.4.286 ACCESS-CLASS restricted )
  ibmattributetypes=( 1.3.18.0.2.4.287 ACCESS-CLASS system )
  ibmattributetypes=( 2.5.4.1 ACCESS-CLASS normal )
  ibmattributetypes=( 1.3.6.1.4.1.1466.101.120.6 ACCESS-CLASS normal )
  ibmattributetypes=( 2.5.21.5 ACCESS-CLASS system )
  ibmattributetypes=( 2.16.840.1.113730.3.1.5 ACCESS-CLASS normal )
  ibmattributetypes=( 2.16.840.1.113730.3.1.8 ACCESS-CLASS sensitive )
  ibmattributetypes=( 2.16.840.1.113730.3.1.77 ACCESS-CLASS normal )
  ibmattributetypes=( 1.3.6.1.4.1.1466.101.120.7 ACCESS-CLASS normal )
  ibmattributetypes=( 2.16.840.1.113730.3.1.7 ACCESS-CLASS normal )
  ibmattributetypes=( 2.5.4.3 ACCESS-CLASS normal )
  ibmattributetypes=( 2.5.18.1 ACCESS-CLASS system )
  ibmattributetypes=( 2.5.18.3 ACCESS-CLASS system )
  ibmattributetypes=( 2.16.840.1.113730.3.1.10 ACCESS-CLASS normal )
  ibmattributetypes=( 2.5.21.2 ACCESS-CLASS system )
  ibmattributetypes=( 2.5.21.1 ACCESS-CLASS system )
```

```
ibmattributetypes=( 2.5.4.49 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.288 ACCESS-CLASS restricted )
ibmattributetypes=( 1.3.18.0.2.4.470 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2244 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2243 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.1155 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.1780 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2242 ACCESS-CLASS sensitive )
ibmattributetypes=( 0.9.2342.19200300.100.1.24 ACCESS-CLASS system )
ibmattributetypes=( 0.9.2342.19200300.100.1.23 ACCESS-CLASS system )
ibmattributetypes=( 1.3.6.1.4.1.1466.101.120.16 ACCESS-CLASS system )
ibmattributetypes=( 2.5.21.4 ACCESS-CLASS system )
ibmattributetypes=( 2.5.21.8 ACCESS-CLASS system )
ibmattributetypes=( 2.5.4.31 ACCESS-CLASS normal )
ibmattributetypes=( 2.16.840.1.113730.3.1.198 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.18.4 ACCESS-CLASS system )
ibmattributetypes=( 2.5.18.2 ACCESS-CLASS system )
ibmattributetypes=( 2.5.4.41 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.21.7 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.1466.101.120.5 ACCESS-CLASS normal )
ibmattributetypes=( 2.16.840.1.113730.3.1.9 ACCESS-CLASS normal )
ibmattributetypes=( 2.16.840.1.113730.3.1.11 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.0 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.21.6 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.289 ACCESS-CLASS restricted )
ibmattributetypes=( 1.3.18.0.2.4.290 ACCESS-CLASS system )
ibmattributetypes=( 2.16.840.1.113730.3.1.34 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.299 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.302 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.300 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.298 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.301 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.304 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.303 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.18.10 ACCESS-CLASS system )
ibmattributetypes=( 2.5.18.6 ACCESS-CLASS system )
ibmattributetypes=( 1.3.6.1.4.1.1466.101.120.13 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.1466.101.120.15 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.1466.101.120.14 ACCESS-CLASS normal )
ibmattributetypes=( 2.16.840.1.113730.3.1.6 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.50 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.35 ACCESS-CLASS critical )
objectclasses=( 2.5.6.1 NAME 'alias' SUP top STRUCTURAL MUST ( aliasedObjectName ) )
objectclasses=( 1.3.18.0.2.6.74 NAME 'aliasObject' SUP top AUXILIARY MUST ( aliasedObjectName ) )
objectclasses=( 2.16.840.1.113730.3.2.1 NAME 'changelogentry'
  DESC 'used to represent changes made to a directory server.' SUP top STRUCTURAL
  MUST ( targetdn $ changetime $ changenumber $ changetype )
  MAY ( modifiersname $ changes $ newrdn $ deleteoldrdn $ newsuperior ) )
objectclasses=( 1.3.18.0.2.6.28 NAME 'container' DESC 'An object that can contain other objects.' SUP top
  STRUCTURAL MUST ( cn ) )
objectclasses=( 1.3.18.0.2.6.174 NAME 'ibmsubschema' SUP subschema AUXILIARY MAY ibmattributetypes )
objectclasses=( 1.3.18.0.2.6.262 NAME 'ibm-changelog'
  DESC 'IBM extension to changeLogEntry to include ibm-changeInitiatorsName attribute' SUP TOP
  AUXILIARY MAY ( ibm-changeinitiatorsname ) )
objectclasses=( 2.16.840.1.113730.3.2.6 NAME 'referral' SUP top MUST ref )
objectclasses=( 2.5.20.1 NAME 'subschema' SUP top AUXILIARY MAY ( ditstructurerules $ nameforms $
  ditcontentrules $ objectclasses $ attributetypes $ matchingrules $ matchingruleuse $ ldapsyntaxes ) )
objectclasses=( 2.5.17.0 NAME 'subentry' SUP top STRUCTURAL MUST ( cn $ subtreespecification ) )
objectclasses=( 2.5.6.0 NAME 'top' ABSTRACT MUST objectclass )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.3 DESC  'attribute type description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.5 DESC  'binary' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.7 DESC  'boolean' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.15 DESC  'directory string' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.12 DESC  'dn' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.16 DESC  'dit content rule description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.17 DESC  'dit structure rule description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.24 DESC  'generalized time' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.26 DESC  'ia5 string' )
ldapsyntaxes=( 1.3.18.0.2.8.1 DESC  'ibm attribute type description' )
ldapsyntaxes=( 1.3.18.0.2.8.3 DESC  '16 byte DCE UUID value in hex string representation.
  TimeLow-TimeMid-Version/TimeHi-Variant/ClkSeqHi/ClkSeqLow-NodeID.
  For example, 11fdfa9e-1ec6-11d5-8ace-0006292197c7' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.27 DESC  'integer' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.54 DESC  'ldap syntax description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.30 DESC  'matching rule description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.31 DESC  'matching rule use description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.35 DESC  'name form description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.37 DESC  'object class description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.40 DESC  'octet string' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.38 DESC  'oid' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.58 DESC  'substring assertion' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.50 DESC  'telephone number' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.53 DESC  'utc time' )
```

```
matchingrules=( 2.5.13.13 NAME 'booleanmatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )
matchingrules=( 1.3.6.1.4.1.1466.109.114.1 NAME 'caseexactia5match'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
matchingrules=( 2.5.13.5 NAME 'caseexactmatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
matchingrules=( 2.5.13.6 NAME 'caseexactorderingmatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
matchingrules=( 2.5.13.7 NAME 'caseexactsubstringsmatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.58 )
matchingrules=( 1.3.6.1.4.1.1466.109.114.2 NAME 'caseignoreia5match'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
matchingrules=( 2.5.13.3 NAME 'caseignoreorderingmatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
matchingrules=( 2.5.13.2 NAME 'caseignorematch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
matchingrules=( 2.5.13.4 NAME 'caseignoresubstringsmatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.58 )
matchingrules=( 2.5.13.1 NAME 'distinguishednamematch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
matchingrules=( 1.3.18.0.2.4.405 NAME 'distinguishednameorderingmatch'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
matchingrules=( 1.3.18.0.2.22.2 NAME 'ibm-entryuuidmatch' SYNTAX 1.3.18.0.2.8.3 )
matchingrules=( 2.5.13.27 NAME 'generalizedtimematch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )
matchingrules=( 2.5.13.28 NAME 'generalizedtimeorderingmatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )
matchingrules=( 2.5.13.14 NAME 'integermatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )
matchingrules=( 2.5.13.29 NAME 'integerfirstcomponentmatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )
matchingrules=( 2.5.13.0 NAME 'objectidentifiermatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )
matchingrules=( 2.5.13.30 NAME 'objectidentifierfirstcomponentmatch'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )
matchingrules=( 2.5.13.17 NAME 'octetstringmatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 )
matchingrules=( 2.5.13.20 NAME 'telephonenumbermatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.50 )
matchingrules=( 2.5.13.21 NAME 'telephonenumbersubstringsmatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.58 )
matchingrules=( 2.5.13.25 NAME 'utctimematch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.53 )
```

Appendix B. SDBM schema

This appendix shows the schema for SDBM assuming the suffix is `sysplex=myRACF`.

```
CN=SCHEMA,sysplex=myRACF
cn=SCHEMA
subtreespecification=NULL
objectclass=TOP
objectclass=SUBSCHEMA
objectclass=SUBENTRY
objectclass=IBMSUBSCHEMA
attributetypes=( 2.5.21.5 NAME 'attributeTypes' EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.3 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.470 NAME 'ibmAttributeTypes' EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.18.0.2.8.1 USAGE directoryOperation )
attributetypes=( 1.3.6.1.4.1.1466.101.120.16 NAME 'ldapSyntaxes' EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.54 USAGE directoryOperation )
attributetypes=( 2.5.21.4 NAME 'matchingRules' EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.30 USAGE directoryOperation )
attributetypes=( 2.5.21.8 NAME 'matchingRuleUse' EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.31 USAGE directoryOperation )
attributetypes=( 2.5.4.0 NAME 'objectClass' EQUALITY objectidentifiermatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )
attributetypes=( 2.5.21.6 NAME 'objectClasses' EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.37 USAGE directoryOperation )
attributetypes=( 2.5.18.6 NAME 'subtreeSpecification'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE USAGE directoryOperation )
attributetypes=( 2.5.21.2 NAME 'ditContentRules' EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.16 USAGE directoryOperation )
attributetypes=( 2.5.21.1 NAME 'ditStructureRules' EQUALITY integerFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.17 USAGE directoryOperation )
attributetypes=( 2.5.21.7 NAME 'nameForms' EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.35 USAGE directoryOperation )
attributetypes=( 2.5.4.41 NAME 'name' EQUALITY caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )
attributetypes=( 2.5.4.3 NAME ( 'cn' 'commonname' ) SUP name )
attributetypes=( 1.3.18.0.2.4.185 NAME 'sysplex' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.186 NAME 'profileType' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.188 NAME 'racfAuthorizationDate' DESC 'Date is displayed in yy.ddd format.' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.189 NAME 'racfOwner' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.190 NAME 'racfInstallationData' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.191 NAME 'racfDatasetModel' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.187 NAME 'racfid' DESC 'Identifies the name of a OS/390 Security Server userid or groupid.'
  EQUALITY caseIgnore2 IA5Match SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.197 NAME 'racfAttributes' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.198 NAME 'racfPassword' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.200 NAME 'racfPasswordChangeDate' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.199 NAME 'racfPasswordInterval' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.201 NAME 'racfProgrammerName' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.202 NAME 'racfDefaultGroup' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.203 NAME 'racfLastAccess' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.214 NAME 'racfSecurityLabel' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.205 NAME 'racfSecurityCategoryList' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.206 NAME 'racfRevokeDate' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.207 NAME 'racfResumeDate' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.208 NAME 'racfLogonDays' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.209 NAME 'racfLogonTime' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.210 NAME 'racfClassName' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.211 NAME 'racfConnectGroupName' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.212 NAME 'racfConnectGroupAuthority' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.213 NAME 'racfConnectGroupUACC' EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.204 NAME 'racfSecurityLevel' EQUALITY caseIgnoreIA5Match
```

```
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.192 NAME 'racfSuperiorGroup' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.193 NAME 'racfGroupNoTermUAC' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.194 NAME 'racfSubGroupName' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.196 NAME 'racfGroupUserAccess' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.195 NAME 'racfGroupUserids' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.215 NAME 'SAFDfpDataApplication' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.216 NAME 'SAFDfpDataClass' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.217 NAME 'SAFDfpManagementClass' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.218 NAME 'SAFDfpStorageClass' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.219 NAME 'racfOmvsGroupId' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.220 NAME 'racfOvmGroupId' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.265 NAME 'racfOmvsUid' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.266 NAME 'racfOmvsHome' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.267 NAME 'racfOmvsInitialProgram' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.280 NAME 'racfOvmUid' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.281 NAME 'racfOvmHome' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.282 NAME 'racfOvmInitialProgram' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.283 NAME 'racfOvmFileSystemRoot' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.221 NAME 'SAFAccountNumber' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.223 NAME 'SAFDestination' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.224 NAME 'SAFHoldClass' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.225 NAME 'SAFJobClass' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.226 NAME 'SAFMessageClass' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.227 NAME 'SAFDefaultLoginProc' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.228 NAME 'SAFLogonSize' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.229 NAME 'SAFMaximumRegionSize' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.230 NAME 'SAFDefaultSysoutClass' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.231 NAME 'SAFUserData' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.232 NAME 'SAFDefaultUnit' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.233 NAME 'SAFTsoSecurityLabel' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.222 NAME 'SAFDefaultCommand' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.237 NAME 'racfOperatorClass' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.236 NAME 'racfOperatorIdentification' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.238 NAME 'racfOperatorPriority' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.239 NAME 'racfOperatorReSignon' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.240 NAME 'racfTerminalTimeout' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.241 NAME 'racfStorageKeyword' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.242 NAME 'racfAuthKeyword' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.243 NAME 'racfMformKeyword' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.244 NAME 'racfLevelKeyword' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.245 NAME 'racfMonitorKeyword' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.246 NAME 'racfRoutcodeKeyword' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.247 NAME 'racfLogCommandResponseKeyword' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.248 NAME 'racfMGIDKeyword' EQUALITY caseIgnoreIA5Match
```

```
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.249 NAME 'racfDOMKeyword' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.250 NAME 'racfKEYKeyword' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.251 NAME 'racfCMDSYSKeyword' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.252 NAME 'racfUDKeyword' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.253 NAME 'racfMscopeSystems' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.254 NAME 'racfAltGroupKeyword' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.255 NAME 'racfAutoKeyword' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.234 NAME 'racfPrimaryLanguage' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.235 NAME 'racfSecondaryLanguage' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.256 NAME 'racfWorkattrUsername' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.257 NAME 'racfBuilding' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.258 NAME 'racfDepartment' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.259 NAME 'racfRoom' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.261 NAME 'racfAddressLine1' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.262 NAME 'racfAddressLine2' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.263 NAME 'racfAddressLine3' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.264 NAME 'racfAddressLine4' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.260 NAME 'racfWorkAttrAccountNumber' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.268 NAME 'racfNetviewInitialCommand' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.269 NAME 'racfDefaultConsoleName' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.270 NAME 'racfCTLKeyword' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.271 NAME 'racfMSGRCVRKeyword' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.272 NAME 'racfNetviewOperatorClass' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.273 NAME 'racfDomains' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.274 NAME 'racfNGMFADMKeyword' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.279 NAME 'racfDCEAutoLogin' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.277 NAME 'racfDCEHomeCell' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.278 NAME 'racfDCEHomeCellUUID' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.276 NAME 'racfDCEPrincipal' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.275 NAME 'racfDCEUUID' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.826 NAME 'racfOmvsMaximumAddressSpaceSize' DESC 'Represents the ASSIZEMAX
    (address-space-size) field of the OMVS RACF SEGMENT.' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.827 NAME 'racfOmvsMaximumCPUTime' DESC 'Represents the CPUTIMEMAX
    (cpu-time) field of the OMVS RACF SEGMENT.' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.828 NAME 'racfOmvsMaximumFilesPerProcess' DESC 'Represents the MMAPAREAMAX
    (memory-map-size) field of the OMVS RACF SEGMENT.' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.829 NAME 'racfOmvsMaximumMemoryMapArea' DESC 'Represents the ASSIZEMAX
    (address-space-size) field of the OMVS RACF SEGMENT.' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.830 NAME 'racfOmvsMaximumProcessesPerUID' DESC 'Represents the PROCUSERMAX
    (processes-per-UID) field of the OMVS RACF SEGMENT.' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.831 NAME 'racfOmvsMaximumThreadsPerProcess' DESC 'Represents the THREADSMAX
    (threads-per-process) field of the OMVS RACF SEGMENT.' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.1099 NAME 'racfLNotesShortName' DESC 'represents the SNAME field of the RACF
    LNOTES segment' EQUALITY caseExactMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.1100 NAME 'racfNDSUserName' DESC 'Represents the UNAME field of the RACF NDS segment' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.1153 NAME 'racfCurKeyVersion' DESC 'Current key version' EQUALITY caseIgnoreMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.1091 NAME 'krbPrincipalName' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.2.840.113556.1.4.77 NAME 'maxTicketAge'
```

```
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.1144 NAME 'racfConnectAttributes' DESC 'RACF Connect Attributes' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.1145 NAME 'racfConnectAuthDate' DESC 'RACF Connect Auth Date' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.1146 NAME 'racfConnectCount' DESC 'RACF Connect Count' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.1147 NAME 'racfConnectLastConnect' DESC 'RACF Connect Last Connect' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.1148 NAME 'racfConnectOwner' DESC 'RACF Connect Owner' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.1149 NAME 'racfConnectResumeDate' DESC 'RACF Connect Resume Date' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.1150 NAME 'racfConnectRevokeDate' DESC 'RACF Connect Revoke Date' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.1151 NAME 'racfGroupId' DESC 'RACF group ID' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.1913 NAME 'racfGroupUniversal' DESC 'RACF universal group indicator' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.2007 NAME 'racfEncryptType' DESC 'RACF encrypt type' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.1152 NAME 'racfUserId' DESC 'RACF userid' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.1162 NAME 'racfLDAPBindDN' DESC 'RACF LDAP Bind DN' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.1163 NAME 'racfLDAPBindPw' DESC 'RACF LDAP Bind Password' EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.1164 NAME 'racfLDAPHost' DESC 'RACF LDAP Host' EQUALITY caseIgnoreMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.2239 NAME 'racfLDAPProf' DESC 'RACF LDAP Profile Name' EQUALITY caseIgnoreMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.2240 NAME 'racfOmvsGroupIdKeyword' DESC 'RACF group OMVS keyword' EQUALITY caseIgnoreMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.2241 NAME 'racfOmvsUidKeyword' DESC 'RACF user OMVS keyword' EQUALITY caseIgnoreMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.3090 NAME 'racfOmvsMemoryLimit' DESC 'Represents the MEMLIMIT(non-shared-memory-size)
    field of the RACF user OMVS segment.' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.3089 NAME 'racfOmvsSharedMemoryMaximum' DESC 'Represents the SHMEMMAX(shared-memory-size)
    field of the RACF user OMVS segment.' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.3091 NAME 'racfPasswordEnvelope' DESC 'Envelope containing user password information'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 SINGLE-VALUE USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.3216 NAME 'racfRslKey' DESC 'Represents the RSLKEY(resource-security-level-key)
    field of the RACF user CICS segment.' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
  attributetypes=( 1.3.18.0.2.4.3215 NAME 'racfTslKey' DESC 'Represents the TSLKEY(transaction-security-level-key)
    field of the RACF user CICS segment.' EQUALITY caseIgnoreIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
  ibmattributetypes=( 1.3.18.0.2.4.470 ACCESS-CLASS normal )
  ibmattributetypes=( 1.3.6.1.4.1.1466.101.120.16 ACCESS-CLASS system )
  ibmattributetypes=( 2.5.21.4 ACCESS-CLASS system )
  ibmattributetypes=( 2.5.21.8 ACCESS-CLASS system )
  ibmattributetypes=( 2.5.4.0 ACCESS-CLASS normal )
  ibmattributetypes=( 2.5.21.6 ACCESS-CLASS system )
  ibmattributetypes=( 2.5.18.6 ACCESS-CLASS system )
  ibmattributetypes=( 2.5.21.2 ACCESS-CLASS system )
  ibmattributetypes=( 2.5.21.1 ACCESS-CLASS system )
  ibmattributetypes=( 2.5.21.7 ACCESS-CLASS normal )
  ibmattributetypes=( 2.5.4.41 ACCESS-CLASS normal )
  ibmattributetypes=( 2.5.4.3 ACCESS-CLASS normal )
  ibmattributetypes=( 1.3.18.0.2.4.185 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.186 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.188 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.189 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.190 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.191 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.187 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.197 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.198 ACCESS_CLASS critical )
  ibmattributetypes=( 1.3.18.0.2.4.200 ACCESS_CLASS critical )
  ibmattributetypes=( 1.3.18.0.2.4.199 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.201 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.202 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.203 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.214 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.205 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.206 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.207 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.208 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.209 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.210 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.211 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.212 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.213 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.204 ACCESS_CLASS critical )
  ibmattributetypes=( 1.3.18.0.2.4.192 ACCESS_CLASS sensitive )
  ibmattributetypes=( 1.3.18.0.2.4.193 ACCESS_CLASS sensitive )
```

```
ibmattributetypes=( 1.3.18.0.2.4.194 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.196 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.195 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.215 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.216 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.217 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.218 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.219 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.220 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.265 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.266 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.267 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.280 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.281 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.282 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.283 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.221 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.223 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.224 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.225 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.226 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.227 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.228 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.229 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.230 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.231 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.232 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.233 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.222 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.237 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.236 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.238 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.239 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.240 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.241 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.242 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.243 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.244 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.245 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.246 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.247 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.248 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.249 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.250 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.251 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.252 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.253 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.254 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.255 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.234 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.235 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.256 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.257 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.258 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.259 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.261 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.262 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.263 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.264 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.260 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.268 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.269 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.270 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.271 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.272 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.273 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.274 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.279 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.277 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.278 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.276 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.275 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.826 ACCESS_CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.827 ACCESS_CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.828 ACCESS_CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.829 ACCESS_CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.830 ACCESS_CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.831 ACCESS_CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.1099 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.1100 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.1153 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.1091 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.2.840.113556.1.4.77 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.1144 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.1145 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.1146 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.1147 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.1148 ACCESS_CLASS sensitive )
```

```
ibmattributetypes=( 1.3.18.0.2.4.1149 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.1150 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.1151 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.1913 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2007 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.1152 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.1162 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.1163 ACCESS_CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.1164 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2239 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2240 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2241 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3090 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3089 ACCESS_CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3091 ACCESS_CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3216 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3215 ACCESS-CLASS sensitive )
objectclasses=( 2.5.6.0 NAME 'top' ABSTRACT MUST ( objectclass ) )
objectclasses=( 2.5.17.0 NAME 'subEntry' SUP top STRUCTURAL MUST ( cn $ subTreeSpecification ) )
objectclasses=( 2.5.20.1 NAME 'subSchema' SUP top AUXILIARY MAY ( ditStructureRules $ nameForms $
  ditContentRules $ objectClasses $ attributeTypes $ matchingRules $ matchingRuleUse $ ldapSyntaxes ) )
objectclasses=( 1.3.18.0.2.6.174 NAME 'ibmSubSchema' SUP subSchema AUXILIARY MAY ( ibmAttributeTypes ) )
objectclasses=( 1.3.18.0.2.6.55 NAME 'racfbase' DESC 'Represents the base of the Directory Information Tree
  that publishes information stored by the OS/390 Security Server RACF service.' SUP top STRUCTURAL MAY ( sysplex ) )
objectclasses=( 1.3.18.0.2.6.56 NAME 'racfProfileType' DESC 'Represents a container below which individual
  RACF profile entries will be published.' SUP top STRUCTURAL MUST ( profileType ) )
objectclasses=( 1.3.18.0.2.6.57 NAME 'racfBaseCommon' DESC 'Represents a common base class for all RACF profiles.'
  SUP top ABSTRACT MAY ( racfOwner $ racfInstallationData $ racfDatasetModel $ racfAuthorizationDate ) )
objectclasses=( 1.3.18.0.2.6.58 NAME 'racfUser' DESC 'Represents a RACF USER Profile entry.' SUP racfbasecommon STRUCTURAL MUST ( racfid )
  MAY ( racfAuthorizationDate $ racfAttributes $ racfPassword $ racfPasswordChangeDate $ racfPasswordEnvelope $ racfPasswordInterval $
  racfProgrammerName $ racfDefaultGroup $ racfLastAccess $ racfSecuritylabel $ racfSecurityCategoryList $ racfRevokeDate $
  racfResumeDate $ racfLogonDays $ racfLogonTime $ racfClassName $ racfConnectGroupName $ racfConnectGroupAuthority $
  racfConnectGroupUACC $ racfSecurityLevel ) )
objectclasses=( 1.3.18.0.2.6.59 NAME 'racfGroup' DESC 'Represents a RACF GROUP Profile entry.' SUP racfbasecommon STRUCTURAL MUST
 ( racfid ) MAY ( racfSuperiorGroup $ racfGroupNoTermUAC $ racfGroupUniversal $ racfSubGroupName $ racfGroupUserAccess $
 racfGroupUserids ) )
objectclasses=( 1.3.18.0.2.6.60 NAME 'SAFDfpSegment' DESC 'Represents the SAF DFP portions of a RACF USER or GROUP profile.'
  SUP top AUXILIARY MAY ( SAFDfpDataApplication $ SAFDfpDataClass $ SAFDfpManagementClass $ SAFDfpStorageClass ) )
objectclasses=( 1.3.18.0.2.6.61 NAME 'racfGroupOmvsSegment' DESC 'Represents the OS/390 OMVS Group information in a RACF GROUP profile.'
  SUP top AUXILIARY MAY ( racfOmvsGroupId $ racfOmvsGroupIdKeyword ) )
objectclasses=( 1.3.18.0.2.6.62 NAME 'racfGroupOvmSegment' DESC 'Represents the OS/390 OVM Group information in a RACF GROUP profile.'
  SUP top AUXILIARY MAY ( racfOvmGroupId ) )
objectclasses=( 1.3.18.0.2.6.63 NAME 'racfUserOmvsSegment' DESC 'Represents the OS/390 OMVS User information in a RACF USER profile.'
  SUP top AUXILIARY MAY ( racfOmvsUid $ racfOmvsHome $ racfOmvsInitialProgram $ racfOmvsMaximumAddressSpaceSize $ racfOmvsMaximumCPUTime $
  racfOmvsMaximumFilesPerProcess $ racfOmvsMaximumMemoryMapArea $ racfOmvsMaximumProcessesPerUID $ racfOmvsMaximumThreadsPerProcess $
  racfOmvsMemoryLimit $ racfOmvsSharedMemoryMaximum $ racfOmvsUidKeyword ) )
objectclasses=( 1.3.18.0.2.6.64 NAME 'racfUserOvmSegment' DESC 'Represents the OS/390 OVM User information protion of a RACF USER profile.'
  SUP top AUXILIARY MAY ( racfOvmUid $ racfOvmHome $ racfOvmInitialProgram $ racfOvmFileSystemRoot ) )
objectclasses=( 1.3.18.0.2.6.65 NAME 'SAFTsoSegment' DESC 'Represents the OS/390 TSO information in a RACF USER profile.' SUP top
  AUXILIARY MAY ( SAFAccountNumber $ SAFDestination $ SAFHoldClass $ SAFJobClass $ SAFMessageClass $ SAFDefaultLoginProc $ SAFLogonSize $
  SAFMaximumRegionSize $ SAFDefaultSysoutClass $ SAFUserData $ SAFDefaultUnit $ SAFTsoSecurityLabel $ SAFDefaultCommand ) )
objectclasses=( 1.3.18.0.2.6.66 NAME 'racfCicsSegment' DESC 'Represents the OS/390 CICS information in a RACF USER profile.' SUP top
  AUXILIARY MAY ( racfOperatorClass $ racfOperatorIdentification $ racfOperatorPriority $ racfOperatorReSignon $ racfRslKey $
  $ racfTerminalTimeout $ racfTslKey ) )
objectclasses=( 1.3.18.0.2.6.67 NAME 'racfOperparmSegment' DESC 'Represents the OS/390 Operator parameters in a RACF USER profile.'
  SUP top AUXILIARY MAY ( racfStorageKeyword $ racfAuthKeyword $ racfMformKeyword $ racfLevelKeyword $ racfMonitorKeyword
  $ racfRoutcodeKeyword $ racfLogCommandResponseKeyword $  racfMGIDKeyword $ racfDOMKeyword $ racfKEYKeyword $ racfCMDSYSKeyword $
  racfUDKeyword $ racfMscopeSystems $ racfAltGroupKeyword $ racfAutoKeyword ) )
objectclasses=( 1.3.18.0.2.6.68 NAME 'racfLanguageSegment' DESC 'Represents the OS/390 language information in a RACF USER profile.' SUP top
   AUXILIARY MAY ( racfPrimaryLanguage $ racfSecondaryLanguage ) )
objectclasses=( 1.3.18.0.2.6.69 NAME 'racfWorkAttrSegment' DESC 'Represents the OS/390 work attributes information in a RACF USER profile.'
  SUP top AUXILIARY MAY ( racfWorkAttrUserName $ racfBuilding $ racfDepartment $ racfRoom $ racfAddressLine1 $ racfAddressLine2 $
  racfAddress Line3 $ racfAddressLine4 $ racfWorkAttrAccountNumber ) )
objectclasses=( 1.3.18.0.2.6.70 NAME 'racfNetviewSegment' DESC 'Represents the OS/390 Netview information in a RACF USER profile.' SUP top
   AUXILIARY MAY ( racfNetviewInitialCommand $ racfDefaultConsoleName $ racfCTLKeyword $ racfMSGRCVRKeyword $ racfNetviewOperatorClass $
   racfDomains $ racfNGMFADMKeyword ) )
objectclasses=( 1.3.18.0.2.6.71 NAME 'racfDCESegment' DESC 'Represents the OS/390 DCE segment information in a RACF USER profile.'
  SUP top AUXILIARY MAY ( racfDCEAutoLogin $ racfDCEHomeCell $ racfDCEHomeCellUUID $ racfDCEPrincipal $ racfDCEUUID ) )
objectclasses=( 1.3.18.0.2.6.248 NAME 'racfLNotesSegment' DESC 'Represents the OS/390 LNOTES segment information in a RACF USER profile'
  SUP top AUXILIARY MAY ( racfLNotesShortName ) )
objectclasses=( 1.3.18.0.2.6.249 NAME 'racfNDSSegment' DESC 'Represents the OS/390 NDS segment information in a RACF USER profile'
  SUP top AUXILIARY MAY ( racfNDSUserName ) )
objectclasses=( 1.3.18.0.2.6.260 NAME 'racfKerberosInfo' DESC 'Kerberos information for RACF'
  SUP top AUXILIARY MAY ( krbPrincipalName $ maxTicketAge $ racfCurKeyVersion $ racfEncryptType ) )
objectclasses=( 1.3.18.0.2.6.259 NAME 'racfConnect' DESC 'RACF Connect' SUP top STRUCTURAL MUST ( racfUserId $ racfGroupId )
  MAY ( racfConnectAttributes $ racfConnectAuthDate $ racfConnectCount $ racfConnectGroupAuthority $
  racfConnectGroupUACC $ racfConnectLastConnect $ racfConnectOwner $ racfConnectResumeDate $ racfConnectRevokeDate ) )
objectclasses=( 1.3.18.0.2.6.267 NAME 'racfProxySegment' DESC 'RACF Proxy segment'
  SUP top AUXILIARY MAY ( racfLDAPBindDN $ racfLDAPBindPw $ racfLDAPHost ) )
objectclasses=( 1.3.18.0.2.6.447 NAME 'racfEIMSegment' DESC 'RACF EIM segment'
  SUP top AUXILIARY MAY ( racfLDAPProf ) )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.3 DESC 'attribute type description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.5 DESC 'binary' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.12 DESC 'dn' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.15 DESC 'directory string' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.16 DESC 'dit content rule description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.17 DESC 'dit structure rule description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.26 DESC 'ia5 string' )
```

```
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.27 DESC 'integer' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.30 DESC 'matching rule description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.31 DESC 'matching rule use description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.35 DESC 'name form description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.37 DESC 'object class description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.38 DESC 'oid' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.54 DESC 'ldap syntax description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.58 DESC 'substring assertion' )
ldapsyntaxes=( 1.3.18.0.2.8.1 DESC 'ibm attribute type description' )
matchingrules=( 2.5.13.1 NAME 'distinguishedNameMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
matchingrules=( 2.5.13.2 NAME 'caseIgnoreMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
matchingrules=( 2.5.13.4 NAME 'caseIgnoreSubstringsMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.58 )
matchingrules=( 1.3.6.1.4.1.1466.109.114.2 NAME 'caseIgnoreIA5Match' SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
matchingrules=( 2.5.13.5 NAME 'caseExactMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
matchingrules=( 1.3.6.1.4.1.1466.109.114.1 NAME 'caseexactia5match' SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
matchingrules=( 2.5.13.14 NAME 'integerMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )
matchingrules=( 2.5.13.29 NAME 'integerFirstComponentMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )
matchingrules=( 2.5.13.0 NAME 'objectIdentifierMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )
matchingrules=( 2.5.13.30 NAME 'objectIdentifierFirstComponentMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )
```

Appendix C. SPUFI files

This appendix shows the following SPUFI files:
- "The TDBMDB SPUFI file"
- "The TDBMINDX SPUFI file" on page 456

Note: The same SPUFI files are used to generate both a TDBM and a GDBM database.

The TDBMDB SPUFI file

```
--************************************************************
--*
--* Licensed Materials - Property of IBM
--* 5694-A01
--* (C) Copyright IBM Corp.  2000
--*
--************************************************************

-- Use the following statements to create your LDAP Server DB2 database
-- and tablespaces in SPUFI.  The database and tablespace names you
-- create will be used to update the database section of the LDAP
-- Server configuration file.  You also need to make DB2 decisions,
-- in terms of buffer pool size selection for tablespaces and column
-- size selection, all of which will be directly related to the data that
-- will be stored in the database.  See the instructions below for
-- more information.
--
-- *************************
-- Database Name Information
-- *************************
-- Change -DDDDDDDD- to the name of the LDAP database name you want to create.
-- Be sure this name is updated to match what is defined for databasename in
-- the server configuration file.
--
-- *************************
-- DataBase Owner Information
-- *************************
-- Change the -UUUUUUUU- to the MVS database owner id. This ID will be the
-- highlevel qualifier for the tables
--
-- ***********************
-- Tablespace Information
-- ***********************
--
--   ***********************************************************************
--   NOTE: Refer to the DB2 manuals for a complete listing of valid buffer
--         pool names.
--   ***********************************************************************
--
-- Change the -AAAAAAAA- to the LDAP entry tablespace name you want to create.
--
-- Change the -BBBB- to the buffer pool name for the LDAP entry tablespace.
--     The size of the buffer pool can be determined with the formula:
--
--      result = 62 bytes + <dn column trunc size (from below)> +
--                   <maximum full size of a DN (from below)> +
--       <size of entry data (which includes creator's DN and modifiers DN)>
--
--     There is also a concept of a "spill over" table, where if the entry
--     data does not fit into the row size, it will be broken up in order
--     to fit into a row.  Entry data may be spread across multiple rows
--     if needed.  So in the above formula, the <size of entry data>
--     does not need to be the maximum size of the data, maybe the median
--     size of the data would be a better choice.  See the long entry
```

```
--     tablespace description below.
--
--     The default suggested size is 4K.
--
-- Change the -CCCCCCCC- to the LDAP long entry tablespace name you want to
-- create.
--
-- Change the -DDDD- to the buffer pool name for the LDAP long entry
--     tablespace.  The long entry table space will hold "spill over" rows
--     for entry data that does not fit into the entry table tablespace.
--     To minimize the number of spill over rows, choose a large buffer
--     pool size.
--
--     The default suggested size is 4K.
--
-- Change the -EEEEEEEE- to the LDAP long attribute tablespace name you want to
--     create.
--
-- Change the -FFFF- to the buffer pool name for the LDAP long attribute
--     tablespace.  The long attribute table space will hold "spill over" rows
--     for attribute data that does not fit into the entry table tablespace.
--     To minimize the number of spill over rows, choose a large buffer
--     pool size.
--
--     The default suggested size is 4K.
--
-- Change the -GGGGGGGG- to the LDAP miscellaneous tablespace name you want
--     to create.
--
-- Change the -HHHHHHHH- to the LDAP search tablespace name you want to create.
--
-- Change the -IIII- to the buffer pool name for the LDAP search tablespace.
--     The size of the buffer pool can be determined with the simple formula:
--
--    result = 16 bytes + <search column trunc size (from below)> +
--             <maximum size of attribute value you would like to search for>
--
--     The result value is the maximum number of bytes a row in the search
--     table containing an attribute value will occupy.  Choose a buffer pool
--     size which will accomodate this size.
--
--     The default suggested size is 4K.
--
-- Change the -JJJJJJJJ- to the LDAP replica tablespace name you want to
--     create.
--
-- Change the -KKKKKKKK- to the LDAP descendants tablespace name you want
--     to create.
--
-- ********************************
-- Column Size Selection Information
-- ********************************
-- All searchable attributes of a given entry will be stored in two forms.
-- The first will be a truncated version, which will be used as part of
-- a DB2 index.  The second version will be the entire attribute value,
-- potentially truncated by the buffer pool size you choose.  The reason
-- two versions are stored is so that LDAP/DB2 can use indexes to increase
-- search performance.  The reason we do not index the entire searchable
-- attribute value is because the cost (in terms of DASD) associated with
-- having indexes on a large column where there is a large amount of data.
--
-- The choice of the search column trunc size should take into account system
-- limits you may have (as described in the above), and should account
-- for the typical size of the attribute values that are stored in
-- LDAP.  For example, if most of your data is only 20 bytes long,
-- choosing 20 for this trunc size would be wise.
--
```

```
-- Change -TTTT- to the search column trunc size you determine best fits your
-- attribute data.
--
--     The default suggested size is 32.
--
-- Another search performance enhancement is related to the DN attribute.
-- The DN attribute value is stored separately from the entry data to allow
-- a fast path lookup.  It is also stored in two versions as well.  The
-- reasons are similar to those mentioned above for the attribute column.
-- Since the DN data is stored in it's own column, you need to define the
-- maximum DN attribute value size here.  You also need to choose a dn
-- column trunc size that best fits your data.
--
-- Change -MMMM- to the dn trunc size you determine best fits your dn data.
--
--     The default suggested size is 32.
--
-- Change -NNNN- to the maximum size of a DN.
--
--     The default suggested size is 512.
--
--
-- *************************
-- Storage Group Information
-- *************************
-- Change the -SSSSSSSS- to the storage group you want to contain the
-- LDAP DB2 tablespaces.  Use SYSDEFLT to choose the default storage group.
--   NOTE: The values provided below for PRIQTY and SECQTY probably need
--         to be modified depending on the projected size of the
--         Directory information to be stored.
--

-- ***************************************************************************
-- Use the following statements if you need to delete your LDAP Server DB2
-- database and tablespaces in SPUFI.  You need to remove the '--'
-- from each line before you can run these statements.
-- Change the -AAAAAAAA- to the LDAP entry tablespace name you want to delete.
-- Change the -CCCCCCCC- to the LDAP long entry tablespace name you want to
--                       delete.
-- Change the -EEEEEEEE- to the LDAP long attr tablespace name you want to
--                       delete.
-- Change the -GGGGGGGG- to the LDAP miscellaneous tablespace name you want
--                        to delete.
-- Change the -HHHHHHHH- to the LDAP search tablespace name you want to delete.
-- Change the -JJJJJJJJ- to the LDAP replica tablespace name you want to
--                       delete.
-- Change the -KKKKKKKK- to the LDAP descendants tablespace name you want
--                       to delete.
-- Change the -DDDDDDDD- to the LDAP database name you want to delete.
-- ***************************************************************************

--DROP TABLESPACE -DDDDDDDD-.-AAAAAAAA-;
--DROP TABLESPACE -DDDDDDDD-.-CCCCCCCC-;
--DROP TABLESPACE -DDDDDDDD-.-EEEEEEEE-;
--DROP TABLESPACE -DDDDDDDD-.-GGGGGGGG-;
--DROP TABLESPACE -DDDDDDDD-.-HHHHHHHH-;
--DROP TABLESPACE -DDDDDDDD-.-JJJJJJJJ-;
--DROP TABLESPACE -DDDDDDDD-.-KKKKKKKK-;
--DROP DATABASE   -DDDDDDDD-;
--COMMIT;

-- ************************
-- Create the LDAP database
-- ************************
CREATE DATABASE -DDDDDDDD- STOGROUP -SSSSSSSS-;

-- *******************************
```

```
-- Create the LDAP entry tablespace
-- ********************************
CREATE TABLESPACE -AAAAAAAA- IN -DDDDDDDD-
       USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 7200
       BUFFERPOOL -BBBB-;

-- ************************************
-- Create the LDAP long entry tablespace
-- ************************************
CREATE TABLESPACE -CCCCCCCC- IN -DDDDDDDD-
       USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 7200
       BUFFERPOOL -DDDD-;

-- ************************************
-- Create the LDAP long attr tablespace
-- ************************************
CREATE TABLESPACE -EEEEEEEE- IN -DDDDDDDD-
       USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 7200
       BUFFERPOOL -FFFF-;

-- ****************************
-- Create the LDAP 4K tablespace
-- ****************************
CREATE TABLESPACE -GGGGGGGG- IN -DDDDDDDD-
       SEGSIZE 4
       USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 7200
       BUFFERPOOL BP0;

-- ********************************
-- Create the LDAP search tablespace
-- ********************************
CREATE TABLESPACE -HHHHHHHH- IN -DDDDDDDD-
       USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 7200
       BUFFERPOOL -IIII-;

-- ********************************
-- Create the LDAP replica tablespace
-- ********************************
CREATE TABLESPACE -JJJJJJJJ- IN -DDDDDDDD-
       USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 7200
       BUFFERPOOL -BBBB-;

-- ********************************
-- Create the LDAP descendants tablespace
-- ********************************
CREATE TABLESPACE -KKKKKKKK- IN -DDDDDDDD-
       USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 7200
       BUFFERPOOL BP0;

-- *********************
-- Create the DB2 tables
-- *********************

-- ***************************
-- Create the DIR_ENTRY table
-- ***************************
CREATE TABLE -UUUUUUUU-.DIR_ENTRY (
    EID                 DECIMAL(15,0)      NOT NULL,
    PEID                DECIMAL(15,0),
    ENTRY_SIZE          INTEGER,
    LEVEL               INTEGER,
    ACLSRC              DECIMAL(15,0),
    ACLPROP             CHAR(1),
    OWNSRC              DECIMAL(15,0),
    OWNPROP             CHAR(1),
    CREATE_TIMESTAMP    TIMESTAMP,
    MODIFY_TIMESTAMP    TIMESTAMP,
```

```
    DN_TRUNC            CHAR(-MMMM-)  FOR BIT DATA,
    DN                  VARCHAR(-NNNN-)  FOR BIT DATA,
    ENTRYDATA           LONG VARCHAR       FOR BIT DATA,
    PRIMARY KEY( EID ) )
IN -DDDDDDDD-.-AAAAAAAA-;

-- ******************************
-- Create the DIR_LONGENTRY table
-- ******************************
CREATE TABLE -UUUUUUUU-.DIR_LONGENTRY (
    EID                 DECIMAL(15,0)     NOT NULL,
    SEQ                 INTEGER           NOT NULL,
    ENTRYDATA           LONG VARCHAR      FOR BIT DATA,
    PRIMARY KEY( EID, SEQ ) )
IN -DDDDDDDD-.-CCCCCCCC-;

-- ******************************
-- Create the DIR_LONGATTR table
-- ******************************
CREATE TABLE -UUUUUUUU-.DIR_LONGATTR (
    EID                 DECIMAL(15,0)     NOT NULL,
    ATTR_ID             INTEGER           NOT NULL,
    VALUENUM            INTEGER           NOT NULL,
    SEQ                 INTEGER           NOT NULL,
    ATTRDATA            LONG VARCHAR      FOR BIT DATA,
    PRIMARY KEY( EID, ATTR_ID, VALUENUM, SEQ ) )
IN -DDDDDDDD-.-EEEEEEEE-;

-- ******************************
-- Create the DIR_MISC table
-- ******************************
CREATE TABLE -UUUUUUUU-.DIR_MISC (
    NEXT_EID            DECIMAL(15,0),
    NEXT_ATTR_ID        INTEGER,
    DB_VERSION          CHAR(10),
    DB_CREATE_VERSION   CHAR(10) )
IN -DDDDDDDD-.-GGGGGGGG-;

-- ***************************
-- Create the DIR_CACHE table
-- ***************************
CREATE TABLE -UUUUUUUU-.DIR_CACHE (
    CACHE_NAME          CHAR(25)          NOT NULL,
    MODIFY_TIMESTAMP    TIMESTAMP         NOT NULL,
    PRIMARY KEY( CACHE_NAME, MODIFY_TIMESTAMP ) )
IN -DDDDDDDD-.-GGGGGGGG-;

-- ***************************
-- Create the DIR_ATTRID table
-- ***************************
CREATE TABLE -UUUUUUUU-.DIR_ATTRID (
    ATTR_ID             INTEGER,
    ATTR_NOID           VARCHAR(200)      NOT NULL,
    PRIMARY KEY( ATTR_NOID ) )
IN -DDDDDDDD-.-GGGGGGGG-;

-- ************************
-- Create the DIR_DESC table
-- ************************
CREATE TABLE -UUUUUUUU-.DIR_DESC (
    DEID                DECIMAL(15,0)     NOT NULL,
    AEID                DECIMAL(15,0)     NOT NULL,
    PRIMARY KEY( DEID, AEID ) )
IN -DDDDDDDD-.-KKKKKKKK-;

-- ***************************
-- Create the DIR_SEARCH table
```

```
-- ***************************
CREATE TABLE -UUUUUUUU-.DIR_SEARCH (
    EID                 DECIMAL(15,0)     NOT NULL,
    ATTR_ID             INTEGER           NOT NULL,
    VALUE               CHAR(-TTTT-)      FOR BIT DATA,
    LVALUE              LONG VARCHAR      FOR BIT DATA )
IN -DDDDDDDD-.-HHHHHHHH-;

-- ****************************
-- Create the DIR_REGISTER table
-- ****************************
CREATE TABLE -UUUUUUUU-.DIR_REGISTER (
    ID                  INTEGER           NOT NULL,
    SRV                 VARCHAR(125)      NOT NULL,
    PRIMARY KEY( ID, SRV ) )
IN -DDDDDDDD-.-GGGGGGGG-;

-- ****************************
-- Create the DIR_PROGRESS table
-- ****************************
CREATE TABLE -UUUUUUUU-.DIR_PROGRESS (
    ID                  INTEGER           NOT NULL,
    PRG                 VARCHAR(125)      NOT NULL,
    SRV                 VARCHAR(125)      NOT NULL,
    PRIMARY KEY( ID, PRG, SRV ) )
IN -DDDDDDDD-.-GGGGGGGG-;

-- ***************************
-- Create the DIR_CHANGE table
-- ***************************
CREATE TABLE -UUUUUUUU-.DIR_CHANGE (
    ID                  INTEGER           NOT NULL,
    TYPE                INTEGER           NOT NULL,
    LONGENTRY_SIZE      INTEGER,
    DIN                 VARCHAR(-NNNN-)   NOT NULL,
    LDIF                LONG VARCHAR      NOT NULL,
    PRIMARY KEY( ID ) )
IN -DDDDDDDD-.-JJJJJJJJ-;

-- *****************************
-- Create the DIR_LONGCHANGE table
-- *****************************
CREATE TABLE -UUUUUUUU-.DIR_LONGCHANGE (
    ID                  INTEGER           NOT NULL,
    SEQ                 INTEGER           NOT NULL,
    LDIF                LONG VARCHAR,
    PRIMARY KEY( ID, SEQ ) )
IN -DDDDDDDD-.-JJJJJJJJ-;

-- ***********************************
-- Commit all the above SQL statements
-- ***********************************
COMMIT;
```

The TDBMINDX SPUFI file

```
--************************************************************
--*
--* Licensed Materials - Property of IBM
--* 5694-A01
--* (C) Copyright IBM Corp.  2000
--*
--************************************************************

-- Use the following statements to create your LDAP Server DB2
-- indexes in SPUFI. See the instructions below for more information.
--
```

```
-- *************************
-- DataBase Owner Information
-- *************************
-- Change the -UUUUUUUU- to the MVS database owner id. This ID will be the
-- highlevel qualifier for the tables.  This value should correspond
-- with the value chosen in the LDAP Server DB2 database and tablespace
-- SPUFI script.
--
-- *************************
-- Storage Group Information
-- *************************
-- Change the -SSSSSSSS- to the storage group you want to contain the
-- LDAP DB2 indexes.  Use SYSDEFLT to choose the default storage group.
--   NOTE: The values provided below for PRIQTY and SECQTY probably need
--         to be modified depending on the projected size of the
--         Directory information to be stored.
--
-- *************************
-- Miscellaneous Information
-- *************************
-- All indexes have been defined DEFER YES, which means they need to be
-- recovered at some point.  It is suggested to do the recovery after
-- the database has been populated for databases with large amounts of
-- data.  Use of this option is strictly optional though.
--
-- To not use the DEFER YES option, simply remove DEFER YES globablly.
--

-- ****************************
-- Create the DIR_ENTRY indexes
-- ****************************
CREATE UNIQUE INDEX -UUUUUUUU-.DIR_ENTRYX0 ON -UUUUUUUU-.DIR_ENTRY( EID )
    USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 2160
    DEFER YES;
CREATE INDEX -UUUUUUUU-.DIR_ENTRYX1 ON -UUUUUUUU-.DIR_ENTRY( PEID, EID )
    USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 2160
    DEFER YES;
CREATE INDEX -UUUUUUUU-.DIR_ENTRYX2 ON -UUUUUUUU-.DIR_ENTRY( EID, DN_TRUNC )
    USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 2160
    DEFER YES;
CREATE INDEX -UUUUUUUU-.DIR_ENTRYX3 ON -UUUUUUUU-.DIR_ENTRY( DN_TRUNC, EID )
    USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 2160
    DEFER YES;

-- *********************************
-- Create the DIR_LONGENTRY indexes
-- *********************************
CREATE UNIQUE INDEX -UUUUUUUU-.DIR_LONGENTRYX1
    ON -UUUUUUUU-.DIR_LONGENTRY( EID, SEQ )
    USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 2160
    DEFER YES;

-- *******************************
-- Create the DIR_LONGATTR indexes
-- *******************************
CREATE UNIQUE INDEX -UUUUUUUU-.DIR_LONGATTRX1
    ON -UUUUUUUU-.DIR_LONGATTR( EID, ATTR_ID, VALUENUM, SEQ )
    USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 2160
    DEFER YES;

-- ****************************
-- Create the DIR_CACHE indexes
-- ****************************
CREATE UNIQUE INDEX -UUUUUUUU-.DIR_CACHEX1
    ON -UUUUUUUU-.DIR_CACHE( CACHE_NAME, MODIFY_TIMESTAMP )
    USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 2160
    DEFER YES;
```

```
-- ****************************
-- Create the DIR_ATTRID indexes
-- ****************************
CREATE UNIQUE INDEX -UUUUUUUU-.DIR_ATTRIDX1
    ON -UUUUUUUU-.DIR_ATTRID( ATTR_NOID )
    USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 2160
    DEFER YES;

-- ***************************
-- Create the DIR_DESC indexes
-- ***************************
CREATE UNIQUE INDEX -UUUUUUUU-.DIR_DESCX1
    ON -UUUUUUUU-.DIR_DESC( DEID, AEID )
    USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 2160
    DEFER YES;

-- *****************************
-- Create the DIR_SEARCH indexes
-- *****************************
CREATE INDEX -UUUUUUUU-.DIR_SEARCHX1
    ON -UUUUUUUU-.DIR_SEARCH( ATTR_ID, VALUE, EID )
    USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 2160
    DEFER YES;

CREATE INDEX -UUUUUUUU-.DIR_SEARCHX2
    ON -UUUUUUUU-.DIR_SEARCH( EID, ATTR_ID )
    USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 2160 CLUSTER
    DEFER YES;

-- *******************************
-- Create the DIR_REGISTER indexes
-- *******************************
CREATE UNIQUE INDEX -UUUUUUUU-.DIR_REGISTERX1
    ON -UUUUUUUU-.DIR_REGISTER( ID, SRV )
    USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 2160
    DEFER YES;

-- *******************************
-- Create the DIR_PROGRESS indexes
-- *******************************
CREATE UNIQUE INDEX -UUUUUUUU-.DIR_PROGRESSX1
    ON -UUUUUUUU-.DIR_PROGRESS( ID, PRG, SRV )
    USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 2160
    DEFER YES;

-- *****************************
-- Create the DIR_CHANGE indexes
-- *****************************
CREATE UNIQUE INDEX -UUUUUUUU-.DIR_CHANGEX1 ON -UUUUUUUU-.DIR_CHANGE( ID )
    USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 2160
    DEFER YES;

-- *********************************
-- Create the DIR_LONGCHANGE indexes
-- *********************************
CREATE UNIQUE INDEX -UUUUUUUU-.DIR_LONGCHANGEX1
    ON -UUUUUUUU-.DIR_LONGCHANGE( ID, SEQ )
    USING STOGROUP -SSSSSSSS- PRIQTY 14400 SECQTY 2160
    DEFER YES;

-- ***********************************
-- Commit all the above SQL statements
-- ***********************************
COMMIT;
```

Appendix D. Supported server controls

The sections that follow describe the supported server controls.

authenticateOnly

Name: authenticateOnly

Description: Used on an LDAP bind operation to indicate to the LDAP server that it should not attempt to find any group membership information for the client's bind DN.

Assigned object identifier: 1.3.18.0.2.10.2

Target of control: Server

Control criticality: Critical at client's option

Values: There is no value; the **controlValue** field is absent.

Detailed description: This control is valid when sent on an LDAP client's bind request to the LDAP server. The presence of this control on the bind request overrides alternate DN look-ups, extended group searching, and default group membership gathering, and causes the LDAP server to only authenticate the client's bind DN and not gather group information at all. This control is intended for a client who does not care about group memberships and subsequent complete authorization checking using groups, but is using the bind only for authentication to the LDAP server and fast bind processing.

IbmLDAPProxyControl

Name: IbmLDAPProxyControl

Description: Used to provide bind and connection information on extended operation requests that result in LDAP requests to any LDAP server. It is required on **GetDnForUserid** and **GetPrivileges** extended operation requests.

Assigned object identifier: 1.3.18.0.2.10.6

Target of control: EXOP backend of server

Control criticality: Critical

Values: The following ASN.1 (Abstract Syntax Notation One) syntax describes the BER (Basic Encoding Rules) encoding of the control value.

```
ControlValue ::= SEQUENCE {
       BindInformation        [0]   BindInfo OPTIONAL,
       ConnectInformation     [1]   ConnectInfo OPTIONAL }
 }

Where,
        ConnectInfo ::= LDAPURL

        BindInfo ::= SEQUENCE {
               Bind DN          LDAPDN,
               Auth        AuthenticationChoice
        }

        AuthenticationChoice ::= CHOICE{
               Simple          [0]   OCTET STRING,
               Sasl            [3]   SaslCredentials
        }

        SaslCredentials ::= SEQUENCE {
               Mechanism       LDAPString,
               Credentials     OCTET STRING OPTIONAL
             }
```

For more information on ASN.1 and BER, go to the following Web site:

ftp://ftp.rsa.com/pub/pkcs/ascii/layman.asc

Detailed description: This control provides information on extended operation requests that result in the use of the LDAP client to make LDAP requests to any LDAP server. The EXOP backend uses the connection information and the bind information specified in the control to establish an LDAP connection. Then, using the established connection, it issues additional LDAP requests to the server.

If the ConnectInformation is not specified, the EXOP backend attempts to open a connection to its local host using a default port of 389. Otherwise, it uses the LDAP URL specified to open a connection. The specified URL must have the following form:

ldap[**s**]**://***host*[**:***port*]

where:

– *host* is a DNS-style host name

– *port* is an optional port number

– **ldaps** causes the EXOP backend to open a secure LDAP connection. The LDAP server must be set up to use SSL and it cannot use the **sslKeyRingPWStashFile** option. See “Setting up for SSL/TLS” on page 46 for more information on SSL configuration.

If the BindInfo is not specified, the EXOP backend makes all of its LDAP requests anonymously. Otherwise, it uses the Bind DN and the AuthenticationChoice to bind to the LDAP server specified in the ConnectInformation. The EXOP backend does not support the SASL authentication choice which is described in the ASN.1.

IBMModifyDNRealignDNAttributesControl

- **Name: IBMModifyDNRealignDNAttributesControl**
- **Description:** Used by a client to request that a Modify DN operation be extended to realign attribute values for attributes of type **DistinguishedName**, and other specified attribute types known to contain distinguished names, with the new DN values established by the Modify DN operation for those DNs.
- **Assigned object identifier:** 1.3.18.0.2.10.11
- **Target of control:** Server
- **Control criticality:** Critical at client’s option
- **Values:** There is no value; the **controlValue** field is absent. The following ANSI.1 (Abstract Syntax Notation One) syntax describes the BER (Basic Encoding Rules) encoding of the control value.

```
ControlValue ::= SEQUENCE {
replace BOOLEAN
}
```

  For more information on ANS.1 and BER, go to the following Web site:
  ftp://ftp.rsa.com/pub/pkcs/ascii/layman.asc
- **Detailed description:** This control is valid when sent on a client’s Modify DN request. Distinguished names which are renamed may be embedded in DN-syntax attributes throughout the directory contents. It may be desirable to replace the embedded values with their renamed counterparts (realignment). The presence of this control on the Modify DN request causes the server to realign matching attribute values in all **DistinguishedName** attributes, and all other attribute types whose attribute syntax is *Distinguished Name* (OID 1.3.6.1.4.1.1466.115.121.1.12), as well as in the attribute types of *aclEntry* and *entryOwner*, which are known to contain distinguished names. The server will evaluate whether the bound user has permission to modify the candidate attribute values, as determined by the appropriate access controls and the permissions granted by those access controls to the bound DN. If the permissions granted to the bound DN are sufficient to modify the candidate attribute values, those values will be realigned to match their respective new DN values. If any single access check fails, the entire operation fails, and all changes to the directory associated with the current Modify DN operation are undone.

IBMModifyDNTimelimitControl

- **Name: IBMModifyDNTimelimitControl**
- **Description:** Used by a client to request that a Modify DN operation be abandoned if the specified time limit for that operation has been exceeded.
- **Assigned object identifier:** 1.3.18.0.2.10.10
- **Target of control:** Server
- **Control criticality:** Critical at client's option
- **Values:** The following ANSI.1 (Abstract Syntax Notation One) syntax describes the BER (Basic Encoding Rules) encoding of the control value.

```
ControlValue ::= SEQUENCE {
Time Limit INTEGER
}
```

  For more information on ANS.1 and BER, go to the following Web site: ftp://ftp.rsa.com/pub/pkcs/ascii/layman.asc
- **Detailed description:** This control is valid when sent on a client's Modify DN request. Modify DN operations may be long-running operations if they affect many entries in the directory (for example, if they rename an entry with a subtree containing many subordinate entries), so it may be desirable to limit the duration of the operation. The presence of this control on the Modify DN request causes the operation to be abandoned by the server if the number of seconds specified in the control value is exceeded. When the operation is abandoned, all changes to the directory associated with the Modify DN operation are undone.

manageDsaIT

- **Name: manageDsaIT**
- **Description:** Used on a request to suppress referral processing, thereby allowing the client to manipulate referral objects.
- **Assigned object identifier:** 2.16.840.1.113730.3.4.2
- **Target of control:** Server
- **Control criticality:** Critical
- **Values:** There is no value; the **controlValue** field is absent.
- **Detailed description:** This control is valid when sent on a client's search, compare, add, delete, modify, or modify DN request. The presence of the control indicates that the server should not return referrals or search continuation references to the client. This allows the client to read or modify referral objects.

PersistentSearch

Name: PersistentSearch

Description: This control is used on a search request to request not only the current contents of the directory that match the search request but also any entries that match the search specification in the future.

Assigned object identifier: 2.16.840.1.113730.3.4.3

Target of control: Server

Control criticality: Critical at client's option

Values: The following ASN.1 (Abstract Syntax Notation One) syntax describes the BER (Basic Encoding Rules) encoding of the control value.

```
ControlValue :: = SEQUENCE {
changeTypes INTEGER,
changesOnly BOOLEAN,
returnECs BOOLEAN
}
```

Where,

- changeTypes: a bit field that specifies one or more types of changes the client is interested in: 0x01 for add changes, 0x02 for delete changes, 0x04 for modify changes, and 0x08 for modRDN changes.
- changesOnly: If set to **TRUE**, only changed entries that match the search are returned. If set to **FALSE**, existing entries matching the search are returned, in addition to changed entries that match the search.
- returnECs: If set to **TRUE**, an **entryChangeNotification** control is included when returning a changed entry that matches the search. If set to **FALSE**, the control is not included.

For more information on ANS.1 and BER, go to the following Web site:
ftp://ftp.rsa.com/pub/pkcs/ascii/layman.asc

Detailed description: The control is only valid for search requests and must be included in the controls portion of the search request. Support is provided in the z/OS client to create this control and parse the resultant entries. See **ldap_create_persistentsearch_control** and **ldap_parse_entrychange_control** API functions in the *z/OS Integrated Security Services LDAP Client Programming* for more information. A persistent search consists of two phases. The first phase is optional (it is done if changesOnly is **FALSE**), and consists of searching the directory for entries matching the search specification. The second phase consists of executing the search specification against any modifications that occur in the directory and, if found matching, then sending the search results to the waiting client.

Persistent search is only supported in the TDBM and GDBM backends. The **persistentSearch** configuration option can be used in the backend section of the configuration file to enable or disable persistent search for that backend. See Chapter 8, “Customizing the LDAP server configuration,” on page 53 for more information on the **persistentSearch** configuration option.

Server behavior: The server behaves as described in the specification found at http://www.mozilla.org/directory/ietf-docs/draft-smith-psearch-ldap-01.txt, with the following exceptions:

1. An error is returned if an error occurs during processing of the persistent search request. Section 4.b of the specification indicates that SearchResultsDone message is not returned if a persistent search is requested. This is not recognized in the case of an error.
2. If more than one **PersistentSearchControl** is received per search request, LDAP_PROTOCOL_ERROR is returned.
3. If the requesting client is not bound as adminDN, LDAP_UNWILLING_TO_PERFORM is returned.
4. If persistent search is requested and the deference option was set to something other than LDAP_DEREF_NEVER or LDAP_DEREF_FINDING, LDAP_PROTOCOL_ERROR is returned.
5. If a persistent search request is specified for a suffix that does not exist in the LDAP server configuration file, LDAP_NO_SUCH_OBJECT is returned.
6. If a persistent search request is specified for a suffix that is configured but for a search base that does not exist, no search results are returned until the object is added.
7. The search filter and scope are matched before a delete is done, all other operations are matched afterwards. No search results are returned for entries moved out of the search filter or scope due to modification or rename.
8. The server accepts persistent searches to the schema entry, cn=schema,*suffix*.
9. If a **PersistentSearch** control is included in a search request for a TDBM or GDBM backend that has not enabled persistent search, the search request is rejected with LDAP_UNAVAILABLE_CRITICAL_EXTENTION (0x35) if the control is critical. If the control is not critical, a ’normal’ search is performed (even if changesOnly is **TRUE**).
10. Change log entries trimmed by the LDAP server due to the **changeLogMaxAge** or **changeLogMaxEntries** configuration options are not returned to a persistent search of the change log directory.
11. If the **manageDsaIT** control is not specified with the **PersistentSearch** control and phase one of the search finds a referral, the referral is returned to the client. If the base of the search is equal

to or below a referral, the referral is returned and the persistent search second phase does not occur. During the second phase of persistent search, referral entries are always processed like normal entries, even if the **manageDsaIT** control is not specified on the persistent search.

12. Idle connection time out also affects persistent search connections.
13. **sizeLimit** and **timeLimit** parameters and configuration options are respected only during the first phase of persistent search, when existing entries are searched. An error is returned if either limit is exceeded and the persistent search ends. During the second phase, when changed entries are searched, **sizeLimit** and **timeLimit** are ignored.
14. Only the entry specified in a modRDN request (the target of the rename operation) can be returned during the second phase of the persistent search. Subentries or entries modified as part of the realignment process are not returned.
15. In a multiserver or sysplex environment, a persistent search request must be made directly to each LDAP server in the collection.
16. The SDBM backend does not support persistent search. To be notified of changes to RACF userids such as password changes, request a persistent search of the change log directory. If configured, RACF creates a change log entry when a modification is made to a RACF userid.
17. Operational attributes are returned on persistent searches except the following: **aclEntry, aclPropagate, aclSource, entryOwner, ownerPropagate, ownerSource, ibm-allGroups**, and **ibm-allMembers**.

schemaReplaceByValueControl

Name: schemaReplaceByValueContol

Description: This control is used to determine how the LDAP server will process a modify operation with replace values for the schema entry. This control will overide the **schemaReplaceByValue** configuration option.

Assigned object identifier: 1.3.18.0.2.10.20

Target of control: Server

Control criticality: Critical at client's option

Values: The following syntax describes the BER (Basic Encoding Rules) encoding of the control value.

```
ControlValue ::= SEQUENCE {
SchemaReplaceByValue BOOL
}
```

Detailed description: This control is only respected on a modify with replace operation of the schema entry. If this control is present and the value is set to **TRUE**, then each replace value in the modify operation either replaces the existing value (if one exists) in the schema or is added to the schema (if an existing value does not exist). All other values in the schema remain as they are. If the control is present and the value is set to **FALSE**, all the values for that attribute in the schema are replaced by the ones specified in the modify operation. See the "Updating the schema" on page 179 for more information on how LDAP processes a schema modify with replace operation.

Appendix E. Supported extended operations

The sections that follow describe the supported extended operations. For information on ASN.1 (Abstract Syntax Notation One) and BER (Basic Encoding Rules), go to the following Web site:

`ftp://ftp.rsa.com/pub/pkcs/ascii/layman.asc`

changeLogAddEntry

- **Name: changeLogAddEntryRequest**
- **Description:** Causes the LDAP server to create a change log entry in the change log using information passed to the extended operation. All input values must be in UTF8.
- **Assigned Object Identifier:** 1.3.18.0.2.12.48
- **Values:** The following ASN.1 syntax describes the BER encoding of the request value.

```
RequestValue ::= SEQUENCE {
 version INTEGER,
 applicationID    INTEGER,
 userid  OCTET STRING,
 group  OCTET STRING,
 changeType ENUMERATED {
    add (0),
    delete (1),
    modify (2),
    rename (3) },
 changeTime OCTET STRING,
 initiator OCTET STRING,
 changes SEQUENCE OF  changeAttributeList OPTIONAL}
```

Where,

version ::= Identifies which version of the interface is being used. Currently the only value supported is 1. If the interface is extended in the future then other values will be supported.

applicationID ::= 1 for RACF. Other applications will have different identifiers. The identifier informs the LDAP server which (if any) translations of the data should be done.

userid ::= A string containing the userid that was created, modified, deleted, or renamed. This string is used to form the value of the **targetDN** attribute in the change log entry.

group ::= For the RACF application, a string containing the group that was created, modified, deleted, or renamed. The RACF application can specify a value for both userid and group to indicate that the change is to the connection of that user to that group. This string is used to form the value of the **targetDN** attribute in the change log entry.

changeType ::= An enumerated value indicating the type of change. This is used to form the value of the **changeType** attribute in the change log entry.

changeTime ::= A string of decimal numbers, used to form the **changeTime** attribute in the change log entry. The format of the string is:yyyymmddhhiiss.uuuuuuZ

Where,

yyyy is year, mm is month, dd is day, hh is hour, ii is minutes, ss is seconds, uuuuuu is micro seconds, Z is a character constant meaning that this time is based on ZULU time, also known as GMT.

initiator ::= A string containing the userid that made the change. This string is used to form the value of the **ibm-changeInitiatorsName** attribute in the change log entry.

```
changeAttributeList ::= SEQUENCE {
 field  attributeDescription,
 vals  SET OF AttributeValue,
 action  ENUMERATED {
      add (0),
      replace (1),
      delete (2) },
 requestValue Boolean }
```

Where,

Field is the name of the attribute that has been changed. For RACF, this consists of the segment name followed by a period followed by the field name. LDAP maps the RACF segment and field name to an LDAP attribute name.

vals is a ber representation (length and data) of the new attribute value.

Action describes what has happened to the attribute (value add, replace, or delete). To indicate that an entire attribute is deleted, specify an action of delete with no value in the vals field.

RequestValue is a flag that, if true, indicates that the attribute value in the vals field is not present and should be requested from the application.

The changeAttributeList values are used to form the **changes** attribute in the change log entry. If changeAttributeList is not specified, a change log entry is created without a **changes** attribute. This acts as a notification to the user of the change log that it should read the entire entry out of the directory tree.

- **Response object identifier:** 1.3.18.0.2.12.49
- **Response description:** This response is used to return error information if an invalid **changeLogAddEntryRequest** is passed to the LDAP server. If no errors are encountered, then an indication of success is returned to the caller. All output is in UTF8.
- **Response values:** The following describes the response value.

```
ResponseValue ::= SEQUENCE  {
 changeLogResultCode ENUMERATED {
   success                  (0),
   loggingFailed            (1),
   invalidCredentials       (2),
   remoteNotSupported       (3),
   notConfigured            (4),
   notActive                (5),
   decodeFailed             (6),
   valueOutOfRange          (7),
   dnConvertFailed          (8)
  }
  errorMessage LDAPString
 }
```

- **Response detailed description:**

  The following table summarizes some different error scenarios and the **changeLogAddEntryRequest** response to such scenarios.

Error scenario	changeLogAddEntryRequest's response
An internal error prevents the logging operation from completing	Returns a **loggingFailed** return code
The caller is not in supervisor state	Returns an **invalidCredentials** return code
Change log is not configured	Returns a **notConfigured** return code
Change log is not active	Returns a **notActive** return code
LDAP server is unable to parse the request	Returns a **decodeFailed** return code
Value is outside the range of allowable values	Returns a **valueOutOfRange** return code
LDAP server is unable to convert a RACF userid to an LDAP DN	Returns a **dnConvertFailed** return code

GetDnForUserid

- **Name: GetDnForUserid**

- **Description:** Causes the EXOP backend to open a connection to an LDAP server with Policy Director data to retrieve all of a user ID's distinguished names.
- **Assigned Object Identifier:** 1.3.18.0.2.12.8
- **Values:** The following ASN.1 syntax describes the BER encoding of the request value.

```
RequestValue ::= SEQUENCE {
      Userid       OCTET STRING,
      Searchbase   [0] LDAPDN OPTIONAL,
      EntryTypes   [1] SEQUENCE OF Objectclass-name OPTIONAL
}
```

  Where,

```
Objectclass-name ::= OCTET STRING
```

- **Detailed description:** Given a user ID and the required **IbmLDAPProxyControl**, the EXOP backend opens a connection to the target LDAP server specified in the **IbmLDAPProxyControl** and retrieves all of the specified user ID's distinguished names.

  The search base in the request value establishes the sub-tree to search for the user ID's distinguished names. If this is not specified, the EXOP backend will perform a root DSE search to determine all of the naming contexts of the target LDAP server and proceed to search each naming context for the user ID's distinguished names. In addition to the search base, the user can specify optional object classes to filter distinguished names from the result.
- **Response object identifier:** 1.3.18.0.2.12.10
- **Response description:** Returned by EXOP backend when it receives a **GetDnForUserid** extended operations request.
- **Response values:** The following ASN.1 syntax describes the BER encoding of the response value.

```
ResponseValue ::= SEQUENCE OF Distinguished-name
```

  where,

```
Distinguished-name LDAPDN
```

- **Response detailed description:** When the EXOP backend receives a **GetDnForUserid** extended operation request and the required **IbmLDAPProxyControl**, it issues requests to the target LDAP sever specified in the **IbmLDAPProxyControl** to retrieve all of the specified user ID's distinguished names. The user may further filter the results by specifying a search base and object class names in the request value.

  The following table summarizes some different error scenarios and the EXOP backend's response to such scenarios.

Error scenario	EXOP backend's response
Cannot find distinguished names	Returns an **LDAP_NO_SUCH_OBJECT** return code
Encounters an **LDAP_NO_SUCH_OBJECT** return code in its attempt to bind to the target LDAP server	Returns an **LDAP_INAPPROPRIATE_AUTH** return code
Encounters any other unsuccessful return codes in its attempt to make LDAP requests to the target LDAP server	Returns these return codes encountered and a detailed message describing the point of failure

GetPrivileges

- **Name: GetPrivileges**
- **Description:** Causes the EXOP backend to open a connection to an LDAP server with Policy Director data and retrieve all of a subject's Policy Director privilege information.
- **Assigned Object Identifier:** 1.3.18.0.2.12.7
- **Values:** The following ASN.1 syntax describes the BER encoding of the request value.

```
RequestValue ::= SEQUENCE {
       Subject      LDAPDN,
       DomainName   OCTET STRING OPTIONAL
}
```

- **Detailed description:** Given the subject and the required **IbmLDAPProxyControl**, the EXOP backend opens a connection to the target LDAP server specified in the **IbmLDAPProxyControl** and retrieves all of the specified subject's Policy Director data. If no domain name is specified, the EXOP backend assumes that the subject exists in the DEFAULT domain.
- **Response object identifier:** 1.3.18.0.2.12.9
- **Response description:** Returned by EXOP backend when it receives a **GetPrivileges** extended operations request.
- **Response values:** The following ASN.1 syntax describes the BER encoding of the response value.

```
ResponseValue ::= SEQUENCE {
      DomainName             OCTET STRING,
      SecLoginType           OCTET STRING,
      PrincipalName          OCTET STRING,
      SecPwdValid            BOOLEAN,
      SecAcctValid           BOOLEAN,
      UserUUID               SEQUENCE {
           Username              LDAPDN,
           UserUUID              OCTET STRING
      }
      GroupInfo              SEQUENCE {
           NumberOfGroups        INTEGER,
           GroupUUIDs            SEQUENCE OF SEQUENCE {
                 Groupname           LDAPDN,
                 GroupUUID           OCTET STRING
           }
      }
}
```

- **Response detailed description:** When the EXOP backend receives a **GetPrivileges** extended operation request and the required **IbmLDAPProxyControl**, it issues requests to the target LDAP sever specified in the **IbmLDAPProxyControl** to retrieve all of the subject's Policy Director data. Refer to *Policy Director Authorization Services for z/OS and OS/390 Customization and Use* for descriptions of the fields returned in the response value.

  The following table summarizes some different error scenarios and the EXOP backend's response to such scenarios.

Error scenario	EXOP backend's response
Cannot find any of the fields in the response value	Returns an **LDAP_NO_SUCH_OBJECT** return code
Retrieves more than one UserUUID or more than one GroupUUID per Groupname	Returns an **LDAP_OTHER** return code
Encounters an **LDAP_NO_SUCH_OBJECT** return code in its attempt to bind to the target LDAP server	Returns an **LDAP_INAPPROPRIATE_AUTH** return code
Encounters any other unsuccessful return codes in its attempt to make LDAP requests to the target LDAP server	Returns these return codes encountered and a detailed message describing the point of failure

Start TLS

- **Name: Start TLS Extended Request**
- **Description:** Causes a non-secure connection to change to a secure connection.
- **Assigned Object Identifier:** 1.3.6.1.4.1.1466.20037
- **Values:** None.

- **Detailed description:** The client may send the Start TLS extended request at any time after establishing an LDAP association, except in the following cases:
  - If a secure connection is already established, or
  - During a multi-stage SASL negotiation, or
  - If there are any outstanding LDAP operations on the connection.

  The LDAP server will respond with an indication of whether the change to a secure connection is allowed. If accepted, the client is expected to immediately begin the secure protocol handshake.

  The secure connection may be ended and a non-secure connection resumed by having the client cause a TLS closure alert to be sent to the server. Communication after receiving the TLS closure alert is over a non-secure connection. The client is considered to be in an anonymous authentication state.
- **Response object identifier:** 1.3.6.1.4.1.1466.20037
- **Response description:** Upon receiving the Start TLS extended request, the server will return an extended response containing a response code indicating success or failure.
- **Response values:** For the successful response, no response value is returned. For an error response, a response message indicating the cause of the error is returned.
- **Response detailed description:**

  The following table summarizes some different error scenarios and the server's response to such scenarios.

Error scenario	Server response
Server accepts connection and can handle the request	Returns an **LDAP_SUCCESS** return code
A secure connection is already established	Returns an **LDAP_OPERATIONS_ERROR** return code
Secure connections are not supported by the server	Returns an **LDAP_PROTOCOL_ERROR** return code
There are outstanding operations on the connection	Returns an **LDAP_OPERATIONS_ERROR** return code
A multi-stage SASL negotiation is in progress	Returns an **LDAP_OPERATIONS_ERROR** return code

Appendix F. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer/Library Server versions of z/OS books in the Internet library at:

`www.ibm.com/servers/eserver/zseries/zos/bkserv/`

One exception is command syntax that is published in railroad track format; screen-readable copies of z/OS books with that syntax information are separately available in HTML zipped file form upon request to mhvrcfs@us.ibm.com.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION ″AS IS″ WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact: but do not indicate that it is the legal department.

IBM Corporation
Mail Station P300

2455 South Road
Poughkeepsie, NY 12601-5400
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks of IBM Corporation in the United States or other countries or both:

- AIX
- DATABASE 2
- DB2
- DRDA
- CICS
- IBM
- Lotus
- MVS
- OS/400
- Parallel Sysplex
- RACF
- RETAIN

RMF
SecureWay
S/390
Tivoli
WebSphere
z/OS
z/OS.e

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, and other countries, or both.

Other company, product, or service names may be the trademarks or service marks of others.

Bibliography

This bibliography provides a list of publications that are useful when implementing the LDAP server product.

IBM z/OS SecureWay Security Server publications

- *z/OS Integrated Security Services LDAP Client Programming*, SC24-5924
- *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683
- *z/OS Migration*, GA22-7499
- *z/OS Security Server RACF System Programmer's Guide*, SA22-7681
- *z/OS Security Server RACF Command Language Reference*, SA22-7687
- *z/OS Security Server RACF Callable Services*, SA22-7691
- *z/OS Integrated Security Services Network Authentication Service Administration*, SC24-5926
- *z/OS Integrated Security Services Network Authentication Service Programming*, SC24-5927

IBM C/C++ language publications

- *z/OS C/C++ Programming Guide*, SC09-4765
- *z/OS C/C++ Run-Time Library Reference*, SA22-7821

IBM DB2 publications

- *DB2 ODBC Guide and Reference*, SC18-7423
- *DB2 Application Programming and SQL Guide*, SC18-7415
- *DB2 Installation Guide*, GC18-7418
- *DB2 Command Reference*, SC18-7416
- *DB2 Messages and Codes*, GC18-7422
- *DB2 SQL Reference*, SC18-7426
- *DB2 Data Sharing: Planning and Administration*, SC18-7417
- *DB2 Utility Guide and Reference*, SC18-7427
- *DB2 ODBC Guide and Reference*, SC18-7423

IBM z/OS Cryptographic Service publications

- *z/OS Open Cryptographic Services Facility Application Programming*, SC24-5899
- *z/OS Cryptographic Services ICSF Administrator's Guide*, SA22-7521

Other IBM publications

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Program Directory*, GI10-0670
- *z/OS Crytographic Services System Secure Sockets Layer Programming*, SC24-5901
- *z/OS UNIX System Services Planning*, GA22-7800
- *z/OS Parallel Sysplex Overview*, SA22-7661
- *z/OS MVS Setting Up a Sysplex*, SA22-7625
- *z/OS SDSF Operation and Customization*, SA22-7670
- *z/OS DCE Command Reference*, SC24-5909
- *z/OS and z/OS.e Planning for Installation*, GA22-7504
- *z/OS Introduction and Release Guide*, GA22-7502
- *z/OS DCE Application Development Guide: Directory Services*
- *z/OS Licensed Program Specifications*, GA22-7503
- *z/OS Collection*, SK3T-4269
- *z/OS Collection*
- *Policy Director Authorization Services for z/OS and OS/390 Customization and Use*
- *ServerPac: Installing Your Order*

Glossary

This glossary defines technical terms and abbreviations used in z/OS LDAP documentation. If you do not find the term you are looking for, refer to the index of the appropriate z/OS LDAP manual or view *IBM Dictionary of Computing*, available from

`www.ibm.com/ibm/terminology`

This glossary includes terms and definitions from:

- *IBM Dictionary of Computing*, SC20-1699.
- *Information Technology—Portable Operating System Interface (POSIX),* from the POSIX series of standards for applications and user interfaces to open systems, copyrighted by the Institute of Electrical and Electronics Engineers (IEEE).
- *American National Standard Dictionary for Information Systems,* ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.
- *Information Technology Vocabulary,* developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1.SC1).
- *CCITT Sixth Plenary Assembly Orange Book, Terms and Definitions* and working documents published by the International Telecommunication Union, Geneva, 1978.
- Open Software Foundation (OSF).

A

access control. Ensuring that the resources of a computer system can be accessed only by authorized users in authorized ways.

access control list (ACL). Data that controls access to a protected object. An ACL specifies the privilege attributes needed to access the object and the permissions that may be granted, to the protected object, to principals that possess such privilege attributes.

ACL. Access control list.

attribute. Information of a particular type concerning an object and appearing in an entry that describes the object in the directory information base (DIB). It denotes the attribute's type and a sequence of one or more attribute values, each accompanied by an integer denoting the value's syntax.

B

backend. A subsystem of the LDAP server which implements access to a persistent storage mechanism for information.

C

certificate. Used to prove your identity. A secure server must have a certificate and a public-private key pair. A certificate is issued and signed by a Certificate Authority (CA).

cipher. A method of transforming text in order to conceal its meaning.

CKDS. Cryptographic Key Data Set.

client. A computer or process that accesses the data, services, or resources of another computer or process on the network. Contrast with *server*.

configuration. The manner in which the hardware and software of an information processing system are organized and interconnected.

Cryptographic Key Data Set (CKDS) . (1) A data set that contains the encrypting keys used by an installation. (2) In ICSF, a VSAM data set that contains all the cryptographic keys. Besides the encrypted key value, an entry in the cryptographic key data set contains information about the key.

cryptography. (1) The transformation of data to conceal its meaning. (2) In computer security, the principles, means, and methods for encrypting plaintext and decrypting ciphertext. (3) In ICSF, the use of cryptography is extended to include the generation and verification of MACs, the generation of MDCs and other one-way hashes, the generation and verification of PINs, and the generation and verification of digital signatures.

D

daemon. A long-lived process that runs unattended to perform continuous or periodic system-wide functions such as network control. Some daemons are triggered automatically to perform their task; others operate periodically.

Data Encryption Standard (DES). In computer security, the National Institute of Standards and Technology (NIST) Data Encryption Standard, adopted by the U.S. government as Federal Information Processing Standard (FIPS) Publication 46, which allows only hardware implementations of the data encryption algorithm.

data hierarchy. A data structure consisting of sets and subsets such that every subset of a set is of lower rank than the data of the set.

data model. (1) A logical view of the organization of data in a database. (2) In a database, the user's logical view of the data in contrast to the physically stored data, or storage structure. (3) A description of the organization of data in a manner that reflects information structure of an enterprise.

database. A collection of data with a given structure for accepting, storing, and providing, on demand, data for multiple users.

Database 2 (DB2) . An IBM relational database management system.

DB2. Database 2.

DES. Data Encryption Standard (DES).

directory. (1) A logical unit for storing entries under one name (the directory name) in a CDS namespace. Each physical instance of a directory is called a replica. (2) A collection of open systems that cooperates to hold a logical database of information about a set of objects in the real world.

directory schema. The set of rules and constraints concerning directory information tree (DIT) structure, object class definitions, attribute types, and syntaxes that characterize the directory information base (DIB).

directory service. The directory service is a central repository for information about resources in a distributed system.

distinguished name (DN). One of the names of an object, formed from the sequence of RDNs of its object entry and each of its superior entries.

DN. Distinguished name.

E

environment variable. A variable included in the current software environment that is available to any called program that requests it.

I

ICSF. Integrated Cryptographic Service Facility.

Integrated Cryptographic Service Facility (ICSF). A licensed program that runs under z/OS and provides access to the hardware cryptographic feature for programming applications. The combination of the hardware cryptographic feature and ICSF provides secure high-speed cryptographic services

J

JCL. Job control language.

Job control language (JCL). A control language used to identify a job to an operating system and to describe the job's requirements.

K

key generator utility program (KGUP). A program that processes control statements for generating and maintaining keys in the cryptographic key data set.

KGUP. Key generator utility program.

L

LDAP. Lightweight Directory Access Protocol.

Lightweight Directory Access Protocol (LDAP). A client/server protocol for accessing a directory service.

M

master replica. The first instance of a specific directory in the namespace. After copies of the directory have been made, a different replica can be designated as the master, but only one master replica of a directory can exist at a time.

MD5. Message Digest 5. A hash algorithm.

MKKF. Make Key File.

MKKF utility. A command-line utility used to create public/private key pairs and certificate requests, receive certificate requests into a key ring, and manage keys in a key ring.

N

NOID. Numeric object identifier

O

object class. An identified family of objects that share certain characteristics. An object class can be specific to one application or shared among a group of applications. An application interprets and uses an entry's class-specific attributes based on the class of the object that the entry describes.

OCSF. Open Cryptographic Services Facility.

ODBC. Open database connectivity.

Open Cryptographic Services Facility (OCSF). A derivative of the IBM Keyworks technology which is an implementation of the Common Data Security Architecture (CDSA) for applications running in the UNIX System Services environment.

z/OS Cryptographic Services. A z/OS offering that supplies a set of interfaces for cryptographic functions.

P

private key. Used for the encryption of data. A secure server keeps its private key secret. A secure server sends clients its public key so they can encrypt data to the server. The server then decrypts the data with its private key.

public key. Used for the encryption of data. A secure server makes its public key widely available so that its clients can encrypt data to send to the server. The server then decrypts the data with its private key.

R

RACF. Resource Access Control Facility.

RDN. Relative distinguished name.

referral. An outcome that can be returned by a directory system agent that cannot perform an operation itself. The referral identifies one or more other directory system agents more able to perform the operation.

relative distinguished name (RDN). A component of a DN. It identifies an entry distinctly from any other entries which have the same parent.

replica. A directory in the CDS namespace. The first instance of a directory in the namespace is the master replica. See *master replica*.

replication. The making of a shadow of a database to be used by another node. Replication can improve availability and load-sharing.

Resource Access Control Facility (RACF). An IBM licensed program, that provides for access control by identifying and verifying the users to the system, authorizing access to protected resources, and logging the detected unauthorized access to protected resources.

S

SASL. Simple Authentication Security Layer.

schema. See *directory schema*.

Secure Sockets Layer (SSL) security. A facility used to protect LDAP access.

server. On a network, the computer that contains programs, data, or provides the facilities that other computers on the network can access. Contrast with *client*.

SHA. Secure Hash Algorithm. A hash algorithm required for use with the Digital Signature Standard.

Simple Authentication Security Layer (SASL). Refers to a method of binding using authentication information outside the client and server.

SLAPD. A stand-alone LDAP daemon.

SPUFI. SQL Processor Using File Input.

SQL Processor Using File Input (SPUFI). A facility of the TSO attachment subcomponent that enables the DB2I user to run SQL statements without embedding them in an application program.

SQL. Structured Query Language.

SSL. Secure Sockets Layer.

Structured Query Language (SQL). A standardized language for defining and manipulating data in a relational database.

T

thread. A single sequential flow of control within a process.

Time Sharing Option (TSO). An operating system option that provides interactive time sharing from remote terminals.

Transport Layer Security (TLS). A security protocol that provides communication privacy over the Internet. The protocol alllows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. TLS is based upon SSL Version 3.0

TSO. Time Sharing Option.

U

UCS Transformation Format (UTF). The LDAP Version 3 protocol specifies that data is passed between client and server in the UTF-8 character set.

UTF. UCS Transformation Format.

X

X.500. The CCITT/ISO standard for the open systems interconnection (OSI) application-layer directory. It

allows users to register, store, search, and retrieve information about any objects or resources in a network or distributed system.

Index

Special characters

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Readers' Comments — We'd Like to Hear from You

z/OS
Integrated Security Services
LDAP Server
Administration and Use

Publication No. SC24-5923-06

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

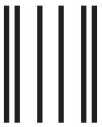
Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC24-5923-06

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY
12601-5400

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Printed in USA

SC24-5923-06