z/OS

IBM Tivoli Directory Server Administration and Use for z/OS

Version 2 Release 2

SC23-6788-01

Note
Before using this information and the product it supports, read the information in "Notices" on page 759.

This edition applies to Version 2 Release 2 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

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.

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

This product includes software developed by NEC Systems Laboratory.

Contents

Figures

Tables

About this document

This document supports z/OS® (5650-ZOS) 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 LDAP administrators. LDAP administrators should be experienced and have previous knowledge of directory services. It is also intended for anyone that will be implementing the directory service.

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

When possible, this information uses cross-document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see *z/OS V2R2 Information Roadmap*.

To find the complete z/OS library, including the z/OS Information Center, see z/OS Internet Library (http://www.ibm.com/systems/z/os/zos/bkserv/).

Internet sources

The following resources are available through the internet to provide additional information about the z/OS library and other security-related topics:

- **Online library**

  To view and print online versions of the z/OS publications, use this address:

  `http://www.ibm.com/systems/z/os/zos/bkserv/`

- **Redbooks®**

  The documents known as IBM® Redbooks that are produced by the International Technical Support Organization (ITSO) are available at the following address:

  `http://www.redbooks.ibm.com`

How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or provide any other feedback that you have.

Use one of the following methods to send your comments:

1. Send an email to mhvrcfs@us.ibm.com.
2. Send an email from the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html).

Include the following information:

- Your name and address.
- Your email address.
- Your telephone or fax number.
- The publication title and order number:
  z/OS V2R2 IBM Tivoli Directory Server Administration and Use for z/OS SC23-6788-01
- The topic and page number that is related to your comment.
- The text of your comment.

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

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

Do not use the feedback methods that are listed for sending comments. Instead, take one of the following actions:

- Contact your IBM service representative.
- Call IBM technical support.
- Visit the IBM Support Portal at z/OS Support Portal (http://www-947.ibm.com/systems/support/z/zos/).

z/OS Version 2 Release 2 summary of changes for IBM Tivoli Directory Server Administration and Use

The following changes are made to z/OS Version 2 Release 2 (V2R2).

New

- "Configuration file options" on page 90 are added:
  - **logFileMicroseconds**
  - **logFileMsgs**
  - **logFileOps**
  - **logFileRecordType**
  - **logFileVersion**
- "Configuration file options" on page 90 contains new information about transition mode added to:
  - **db2Terminate**
  - **db2StartUpRetryLimit**
  - **fileTerminate**
  - **srvStartUpError**
  - **serverCompatLevel**
  - **tcpTerminate**
- **cn=Replication,cn=configuration** entry attributes in "cn=Replication,cn=configuration" on page 141 is updated:
  - **ibm-slapdReplicateSecurityAttributes**
- LDAP server optional command-line parameter, **-t**, transition mode, is added to "Running the LDAP server using the sample JCL" on page 171.
- Transition mode information is added to the XCF command result in "Displaying performance information and server settings" on page 184.
- New information is added to "Activity logging" on page 193 and "Configuring the activity log support" on page 196 for the addition of version 0 and version 1 activity logging.
- New information is added about transition server mode. See "Updating LDAP configurations settings in a sysplex without server outage" on page 210.
- "Replicating password policy operational attributes" on page 383 contains new information about password policy replication.
- New attribute, *ibm-slapdServerCompatibilityLevel=7*, is added to "Root DSE search with base scope" on page 591.
- "User groups considerations in large directories" on page 621 is updated to include new information about large group considerations.
- Appendix C, "Supported server controls," on page 679 has added:
  - **Replication bind failure time stamp control**
- The abandon request has been added in Appendix E, "SMF records," on page 721.
- New information for **logFileRecordType** and **logFileRecordType** are added to Appendix F, "Activity log records," on page 735.
- A new section, Appendix H, "Searching operational attributes," on page 749, is added.

Changed

- "Configuration file options" on page 90 updated:
  - **logFileRolloverDirectory**
  - **logFileRolloverSize**
  - **sslCipherSpecs**
- "Setting up a user ID for your LDAP server" on page 45
- "Setting up for LDBM" on page 61
- "Setting up for CDBM" on page 62
- "Configuring file-based GDBM" on page 64
- "Setting up the security options for the LDAP server" on page 71 is updated to indicate that LDAP no longer supports SSL V2 protocol and that SSL V3, TLS V1.0, TLS V1.1, and TLS V1.2 protocols are supported.
- **cn=configuration** entry attributes in "cn=configuration" on page 139 are updated:
  - **ibm-slapdSortKeyLimit**
  - **ibm-slapdSortSrchAllowNonAdmin**
- "Running the LDAP server using the sample JCL" on page 171 is updated.
- "Environment variables used by the LDAP server" on page 179 are deprecated:
  - **GLDLOG_MICROSECONDS**
  - **GLDLOG_MSG**
  - **GLDLOG_OPS**
  - **GLDLOG_TIME**
- The following utilities are updated:
  - "ds2ldif utility" on page 225
  - "ldif2ds utility" on page 235
- Appendix D, "Supported extended operations," on page 695 updated:
  - **GetEffectiveACL**

Summary of changes for z/OS Version 2 Release 1

See the following publications for all enhancements to z/OS Version 2 Release 1 (V2R1):

- *z/OS V2R2 Migration*
- *z/OS Planning for Installation*
- *z/OS Summary of Message and Interface Changes*
- *z/OS V2R2 Introduction and Release Guide*

Part 1. Administration

Chapter 1. Introducing the LDAP server

The z/OS Lightweight Directory Access Protocol (LDAP) server, part of IBM Tivoli® Directory Server for z/OS (IBM TDS), 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 any Version 2 or Version 3 LDAP directory client
- 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 or hashing
- Password policy
- Basic replication
- Advanced 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®
- Support for sharing directory data in a sysplex
- Plug-in support to extend the LDAP server

This information 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 that is provided by the LDAP server in particular.

z/OS IBM Tivoli Directory Server Client Programming for z/OS describes the LDAP client application programming interfaces (APIs) you can use to develop LDAP applications.

z/OS IBM Tivoli Directory Server Messages and Codes for z/OS describes the messages and reason codes that are issued by the LDAP server.

z/OS IBM Tivoli Directory Server Plug-in Reference for z/OS describes the LDAP server application programming interfaces (APIs) you can use to develop a plug-in for the LDAP server and the ICTX and remote crypto plug-ins provided by the z/OS LDAP server.

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 typically 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 might be able 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, if 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 typically distributed, meaning that the data they contain is spread across many machines, all 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 RFC 1777: *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 13 shows the entire list of supported RFCs.

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

How is information 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 email address. The values depend on what type of attribute it is. For example, a **mail** attribute might contain an email 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.

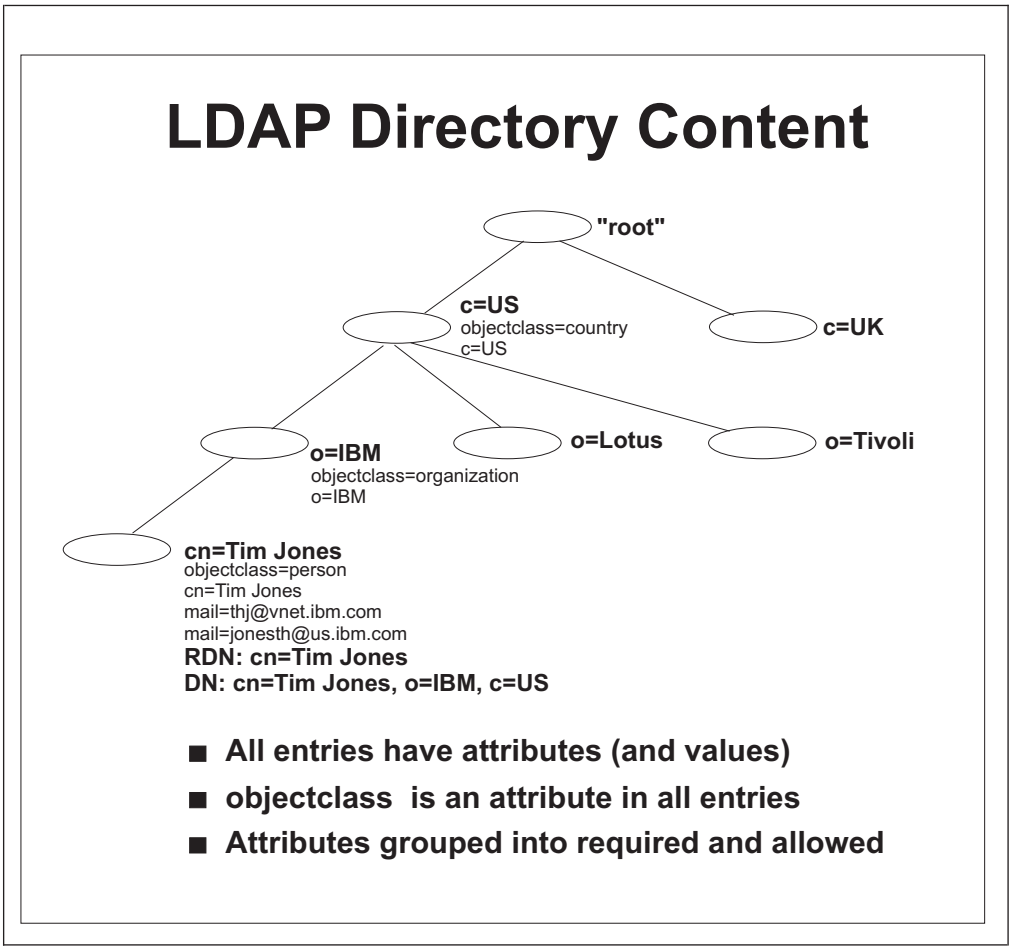


Figure 1. Directory hierarchy example

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 RFC 2253: *Lightweight Directory Access Protocol (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 also be mixed, 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 email 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.

How is the information protected from unauthorized access?

LDAP client requests can be performed using an anonymous identity or the LDAP bind operation can be used to supply an authentication identity. The LDAP server can use the identity to perform authorization checking when accessing entries in the directory. An Access Control List (ACL) provides a means to protect information stored in an LDAP directory. An ACL is used to restrict access to different portions of the directory, to specific directory entries, or to information within an entry. Access control can be specified for individual users or for groups. This authentication process can be used by distributed applications which must implement some form of authentication.

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 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 and store application-specific information.
- **Robust general-purpose databases** The LDAP server comes with TDBM and LDBM backends. There are no restrictions on the types of information that these backends can contain. The TDBM backend is based on DB2® and is a highly scalable database implementation. The LDBM backend keeps its entries in memory for quick access and requires a minimum amount of setup. When the LDAP server is not running, LDBM stores its directory information in z/OS UNIX System Services files.

  Note: DB2 is required to use TDBM, but is not required for LDBM.
- **Access to RACF data**: The LDAP server can be configured to provide read/write access to RACF user, group, connection, and general resource profiles using the LDAP protocol. The LDAP server can also be used to manage RACF options that affect classes. (RACF is a component of the Security Server for z/OS.) If the RACF data is shared across the sysplex, then users, groups, connections, and resource profiles 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 17, "Accessing RACF information," on page 327 for more information.

  Note: To use SDBM for ONLY authentication (LDAP bind processing), any security manager implementing the SAF service required by the RACROUTE REQUEST=VERIFY,ENVIR=CREATE macro can be used. A password or a password phrase can be used for authentication. To use SDBM for accessing and updating user, group, connection, and resource profile information, and to set class options, RACF is required.
- **Configuration backend**: The LDAP server can be configured with a CDBM backend. The CDBM backend is used to store configuration information. See "Setting up for CDBM" on page 62 for more information.
- **Loading and unloading data**: The LDAP server can load many entries into a TDBM directory using the **ldif2ds** utility. See "ldif2ds utility" on page 235 for more information. The LDAP server can also unload many entries from a TDBM, LDBM, or CDBM directory using the **ds2ldif** utility. See "ds2ldif utility" on page 225 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 24, "Using access control," on page 433 about access control and Chapter 23, "Static, dynamic, and nested groups," on page 419 for more information about groups.

- **Threads**: The LDAP server is threaded for optimal performance. A single multi-threaded z/OS LDAP server process handles all incoming requests, reducing the amount of system overhead required.
- **Multiple concurrent servers**: The LDAP server can be configured to permit multiple LDAP servers to serve the TDBM, LDBM, CDBM, or GDBM directory at the same time. The multiple servers might run on the same z/OS image, and they might run on multiple z/OS images in a Parallel Sysplex. This improves availability and might offer improved performance in certain configurations. See "Determining operational mode" on page 146 for more information.
- **Basic replication**: The LDAP server can be configured to maintain replica copies of its database. Master/consumer replication 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 25, "Basic replication," on page 469 for more information. This feature is contrasted with multiple concurrent servers.
- **Advanced replication**: The LDAP server can be configured to act as a supplier, consumer, cascading, or gateway server in an advanced replication environment. An advanced replication environment allows for only certain subtrees or entries in a TDBM, LDBM, or CDBM backend to be replicated to other servers. See Chapter 26, "Advanced replication," on page 487 for more information.
- **Referrals**: The LDAP server is able 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 29, "Referrals," on page 575 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 27, "Alias," on page 555 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, CDBM, or LDBM backend, to the LDAP server schema, or to a RACF user, group, connection, or resource profile. The GDBM backend can be configured to store its entries in DB2 (similar to TDBM) or in z/OS UNIX System Services files (similar to LDBM). See Chapter 28, "Change logging," on page 563 for more information.
- **Configuration**: The LDAP server configuration process can be simplified by using the **dsconfig** 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 dsconfig utility," on page 25 for more information.

  If you do not use the **dsconfig** utility, the LDAP server is highly configurable through a single configuration file that 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 ds.conf file" on page 83 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 various ciphers for encryption to choose from, all that provide server and optionally client authentication by using X.509 certificates. See "Setting up for SSL/TLS" on page 68 for more information.

- **Dynamic workload management**: The LDAP server can be configured to participate in dynamic workload management in a Parallel Sysplex by using TCP/IP connection optimization. With multiple concurrent server instances configured in this way, availability is improved, including resource usage. In addition, performance improvements might be experienced as sysplex resource usage is more evenly balanced across z/OS systems in the sysplex. See "Determining operational mode" on page 146 for more information.
- **Retrieve Policy Director data**: This capability is deprecated. 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 22, "Using extended operations to access Policy Director data," on page 417 for more information.
- **Native authentication**: The z/OS LDAP server allows clients to bind to entries in a TDBM, LDBM, or CDBM 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, LDBM, or CDBM backend along with a security manager-maintained password or password phrase. Password or password phrase authentication is then performed by the security manager. See Chapter 20, "Native authentication," on page 403 for more information.

  Note: To use native authentication, any security manager implementing the SAF service required by the RACROUTE REQUEST=VERIFY,ENVIR=CREATE macro function call can be used.
- **LDAP Version 3 protocol support**: The LDAP server provides support for Version 3 of the LDAP protocol in addition to the LDAP Version 2 protocol. Version 3 includes:
  - All protocol operations
  - Implicit bind
  - Certificate (or Simple Authentication and Security Layer) bind
  - Version 3 referrals
  - Aliases
  - Controls
  - Root DSE support
  - Globalization (UTF-8) support
  - Modify name supported for all entries including subtree move
  - Schema publication
  - Additional syntax support
  - Online schema update capability
- **Dynamic schema**: The LDAP server allows the schema to be changed dynamically through the LDAP protocol. See Chapter 15, "LDAP directory schema," on page 275 for more information.
- **Globalization (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 267 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. To use a certificate on bind, the server must be configured to perform both client and server authentication. This ensures that both entities are who they claim to be. See "Setting up for SSL/TLS" on page 68 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 19, "Kerberos authentication," on page 393 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: *Using Digest Authentication as a SASL Mechanism*) and CRAM-MD5 (RFC 2195: *IMAP/POP AUTHorize Extension for Simple Challenge/Response*) authentication bind methods. See Chapter 21, "CRAM-MD5 and DIGEST-MD5 authentication," on page 413 for more information.
- **Support for root DSE**: The LDAP server supports search operations, including subtree search, against the root of the directory tree as described in RFC 2251: *Lightweight Directory Access Protocol (v3)*. The so-called Root DSE can be accessed using LDAP V3 search operations. See "Root DSE" on page 591 and *z/OS IBM Tivoli Directory Server Client Programming for z/OS* 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 might be a member of static and nested groups in a backend (TDBM, LDBM, or CDBM) 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, "Configuration file options" on page 90 for more information.
- **Supported server controls**: The LDAP server supports the following:
  authenticateOnly
  Do Not Replicate
  IBMLdapProxyControl
  IBMModifyDNRealignDNAttributesControl
  IBMModifyDNTimelimitControl
  IBMSchemaReplaceByValueControl
  manageDsaIT
  No Replication Conflict Resolution
  pagedResults
  PasswordPolicy
  PersistentSearch
  Refresh Entry
  replicateOperationalAttributes
  Replication Supplier ID Bind
  Server Administration
  SortKeyRequest

  See Appendix C, "Supported server controls," on page 679 for more information.
- **Supported extended operations**: The LDAP server supports the following:
  Account status
  Cascading control replication
  changeLogAddEntry
  Control replication
  Control replication error log
  Control replication queue
  Effective password policy
  GetDnForUserid
  GetEffectiveACL
  GetPrivileges
  Quiesce or unquiesce context
  Remote auditing
  Remote authorization
  RemoteCryptoPKCS#11
  RemoteCryptoCCA
  Replication topology
  Start TLS

unloadRequest
User type

See Appendix D, "Supported extended operations," on page 695 for more information.

- **Attribute encryption or hashing**: The LDAP server supports encryption or hashing of the values of several critical attributes to prevent unauthorized access to these attribute values in TDBM, LDBM, and CDBM backends. The attributes that can be encrypted or hashed are as follows:
  ibm-replicaKeyPwd
  ibm-slapdAdminPw
  ibm-slapdMasterPw
  replicaCredentials
  secretKey
  userPassword

  See "Configuring for encryption or hashing" on page 77 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 host name, a specific IPv4 or IPv6 address, or all active and available IPv4 and IPv6 addresses available on the system, along with the port number, can target one or multiple IPv4 or IPv6 interfaces on a system. See the **listen** configuration option, "listen ldap_URL" on page 104, for more information.
- **Persistent search**:The LDAP server provides an event notification mechanism for applications, directories, and meta directories that must maintain a cache of directory information or to synchronize directories when changes are made to an LDAP directory. Persistent search allows these applications to be notified when a change has occurred. See Appendix C, "Supported server controls," on page 679 for more information.
- **ibm-entryuuid attribute**: The LDAP server 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. See "Configuration file options" on page 90 to configure the **serverEtherAddr** option in the LDAP server configuration file.
- **ibm-entryCheckSum and ibm-entryCheckSumOp attributes**: The LDAP server supports the querying of a checksum of all non-operational attributes with the **ibm-entryCheckSum** operational attribute. The LDAP server also supports the **ibm-entryCheckSumOp** operational attribute, which is a checksum of the following operational attributes: **aclEntry, aclPropagate, entryOwner, ownerPropagate, creatorsName, modifiersName, createTimestamp, modifyTimestamp, ibm-entryuuid, ibm-pwdIndividualPolicyDN**, and **ibm-pwdGroupPolicyDN**. The **ibm-entryCheckSum** and **ibm-entryCheckSumOp** operational attributes are used by the **ldapdiff** utility to simplify comparisons of entries between two different servers. See "ldapdiff utility" on page 247 for more information.
- **ibm-allMembers and ibm-allGroups attributes**: The LDAP server supports the querying of the members of static, dynamic, and nested groups in a TDBM, LDBM, or CDBM backend by using 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 attribute.
- **Plug-in support**: The LDAP server can be configured to provide access to extensions to the LDAP server. The extensions are supplied by other products or created by you. Plug-in extensions are invoked before, during, or after the LDAP

server processes a client request. The z/OS LDAP server provides two client operation plug-in extensions of its own: remote crypto and ICTX. See the **plugin** configuration option in Chapter 8, "Customizing the LDAP server configuration," on page 83 for more information about configuring a plug-in extension. See *z/OS IBM Tivoli Directory Server Plug-in Reference for z/OS* for more information about creating a plug-in extension and the remote crypto and ICTX plug-in extensions.

- **Workload Management**: The LDAP server can be configured to use transaction names configured in Workload Manager (WLM) to use different performance goals for LDAP operations based upon the client's IP address or the bound user's distinguished name (DN). See "Workload manager (WLM)" on page 602 for more information about using WLM with the LDAP server.
- **Comparing directories**: The **ldapdiff** utility is provided to compare the directory contents of two different LDAP servers. This utility is useful in determining and optionally synchronizing data between a master and replica server before configuring basic or advanced replication. See "ldapdiff utility" on page 247 for more information.
- **Extended operations utility**: The **ldapexop** utility provides a command line interface for the following extended operations: **Account status, Cascading control replication, Control replication, Control replication error log, Control replication queue, Effective password policy, GetEffectiveACL, User type, Quiesce or unquiesce context**, and **Replication topology**. See "ldapexop utility" on page 255 for more information.
- **Password policy**: The LDAP server supports password policy which is a set of rules that control how passwords are defined, used, and administered. See Chapter 18, "Password policy," on page 365 for more information.
- **Group search limits**: Groups can be used to set specific size and time limits for searches requested by members of the group, providing greater control over usage of LDAP server resources during search operations. See "Managing group search limits" on page 423 for more information.
- **Paged and sorted search results**: The LDAP server supports paged and sorted search results. Paged search results provide paging capabilities for LDAP client applications that want to receive just a subset of search results at a time. Sorted search results enable LDAP client applications to receive sorted search results based on a list of criterion, where each criteria represents a sort key. See "pagedResults" on page 683 and "SortKeyRequest" on page 691 for more information.
- **Administrative group and roles**: The LDAP root administrator can delegate administrative authority to different users by placing them in the administrative group and assigning one or more administrative roles. The roles are assigned in the LDAP administrative group member entry or alternately the roles are assigned in RACF. See Chapter 9, "Administrative group and roles," on page 159 for more information.

Participation in multilevel security

Multilevel security is an enhanced security environment that can be configured on a z/OS system from the SECLABEL class and various SECLABEL-related options. In this environment, the security server and trusted resource managers enforce mandatory access control policies in addition to the typical discretionary access control policies. Resource managers that do not support mandatory access control processing can participate in a multilevel secure system, if they are physically managed, to guarantee that all information made available by that resource manager has the same single security label and all users of the resource manager are permitted to that security label. These servers are referred to as single-level

security. Each LDAP server can participate in a multilevel security environment by being configured as a single-level server. For more information about configuring a z/OS system for multilevel security and how to configure an LDAP server in that environment, see *z/OS Planning for Multilevel Security and the Common Criteria*.

RFCs supported by z/OS LDAP

The z/OS LDAP server supports all or parts of the following Internet Engineering Task Force (IETF) request for comments:

- RFC 1738: *Uniform Resource Locators (URL)*
- RFC 1823: *The LDAP Application Program Interface*
- RFC 2052: *A DNS RR for specifying the location of services (DNS SRV)*
- RFC 2104: *HMAC: Keyed-Hashing for Message Authentication*
- RFC 2195: *IMAP/POP AUTHorize Extension for Simple Challenge/Response*
- RFC 2222: *Simple Authentication and Security Layer (SASL)*
- RFC 2247: *Using Domains in LDAP/X.500 Distinguished Names*
- RFC 2251: *Lightweight Directory Access Protocol (v3)*
- RFC 2252: *Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions*
- RFC 2253: *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*
- RFC 2254: *The String Representation of LDAP Search Filters*
- RFC 2255: *The LDAP URL Format*
- RFC 2256: *A Summary of the X.500 (96) User Schema for use with LDAPv3*
- RFC 2279: *UTF-8, a transformation format of ISO 10646*
- RFC 2373: *IP Version 6 Addressing Architecture*
- RFC 2696: *LDAP Control Extension for Simple Paged Results Manipulation*
- RFC 2713: *Schema for Representing Java(tm) Objects in an LDAP Directory*
- RFC 2714: *Schema for Representing CORBA Object References in an LDAP Directory*
- RFC 2732: *Format for Literal IPv6 Addresses in URLs*
- RFC 2743: *Generic Security Service Application Program Interface Version 2, Update 1*
- RFC 2744: *Generic Security Service API Version 2 : C-bindings*
- RFC 2820: *Access Control Requirements for LDAP*
- RFC 2829: *Authentication Methods for LDAP*
- RFC 2830: *Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security*
- RFC 2831: *Using Digest Authentication as a SASL Mechanism*
- RFC 2849: *The LDAP Data Interchange Format (LDIF)*
- RFC 2891: *LDAP Control Extension for Server Side Sorting of Search Results*
- RFC 3377: *Lightweight Directory Access Protocol (v3): Technical Specification*
- RFC 4517: *LDAP Syntaxes and Matching Rules*
- RFC 4523: *LDAP Schema Definitions for X.509 Certificates*

Although the LDAP V3 protocol RFCs are listed as supported, the specific function that z/OS LDAP supports is listed in the "LDAP Version 3 protocol support" section. See "What are the capabilities of the z/OS LDAP server?" on page 7 for more information.

Note:

1. RFC 2251 is listed as supported, however, the z/OS LDAP server does not support the documented binary option.
2. RFCs 2251 and 2254 are listed as supported, however, the z/OS LDAP server does not support extensible search filters. Approximate search filters are treated as equality search filters in the z/OS LDAP server.

3. RFC 2252 is listed as supported, however, the z/OS LDAP server does not support all documented schema syntaxes. See Chapter 15, "LDAP directory schema," on page 275 for more information about the schema syntaxes supported by the z/OS LDAP server.

LDAP RFCs rely on the International Telecommunication Union (ITU-T) standards to define many of the basic constructs:

X.500 *The Directory: Overview of Concepts, Models and Services*

X.501 *The Directory: Models*

X.520 *The Directory: Selected Attribute Types*

X.690 *Abstract Syntax Notation One (ASN.1)*

Draft RFCs

The z/OS LDAP server supports all or parts of the following request for comment (RFC) draft:

Password Policy for LDAP directories

Superseded RFCs

The following obsolete RFCs were implemented by the z/OS LDAP client and server but have been superseded by current RFCs:

RFC 1777: *Lightweight Directory Access Protocol*
RFC 1778: *The String Representation of Standard Attribute Syntaxes*
RFC 1779: *A String Representation of Distinguished Names*
RFC 1959: *An LDAP URL Format*
RFC 1960: *A String Representation of LDAP Search Filters*

Chapter 2. Planning and roadmap

This topic:

- Shows you where to find information that helps you plan your directory content.
- Contains a roadmap that points you to information that might 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, see Chapter 15, "LDAP directory schema," on page 275 or "Setting up for SDBM" on page 60 for more information.
- How you want to structure your directory data.

  See Chapter 14, "Data model," on page 271 for more information.
- A set of policies for access permissions.

  See Chapter 24, "Using access control," on page 433 for more information.

LDAP server roadmap

Table 1 on page 16 is a roadmap that points you to information that may be helpful in preparing for your LDAP server configuration.

See Chapter 11, "Migrating to z/OS," on page 207 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 V2R2 Program Directory* which comes with the LDAP server tape or cartridge. Be sure to read the license agreement in *z/OS V2R2 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 information. 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 WebSphere® Application Server for z/OS, you must:		
	See the WebSphere documentation for details on LDAP server requirements.	
If you are configuring your LDAP server for z/OS Enterprise Identity Mapping (EIM), you must:		
	See the z/OS EIM 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 **ds.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.	Chapter 11, "Migrating to z/OS," on page 207
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 V2R2 Program Directory*, which contains the complete install instructions. • *z/OS V2R2 Licensed Program Specifications*, which contains the license agreement. • *Memo to Users*, which describes any late changes to the procedures in this information.	
Choose a configuration method from the following options:		
	The **dsconfig** configuration utility uses a profile file as input to generate jobs to set up the system environment and configuration. Check "Restrictions" on page 27 to decide if this method will work for your installation.	Chapter 4, "Configuring an LDAP server using the dsconfig utility," on page 25
	If you do not use the **dsconfig** utility, use the instructions for customizing your configuration.	Chapter 5, "Configuring an LDAP server without the dsconfig utility," on page 39

Chapter 3. Installing and setting up related products

This topic contains information about required and optional products that are necessary to install or set up before configuring the z/OS LDAP server product.

Required products

This section describes the products that must either be installed or configured for the LDAP server to work properly.

To run the LDAP server:	You must:	See:
Workload Manager (WLM)	Install and configure Workload Manager (WLM).	"Installing and setting up WLM (Workload Management)"
Schema backend (based on z/OS UNIX System Services file system).	Ensure that a z/OS UNIX System Services file system exists and has enough space to store the schema backend and that it can be written to by the LDAP server.	"Installing a z/OS UNIX System Services file system for the schema backend" on page 18

Installing and setting up WLM (Workload Management)

The z/OS LDAP server supports Workload Manager (WLM) to allow an installation to set performance goals for work within the LDAP server when compared against other work that is running on the system. The **LDAP** subsystem type is reserved in WLM to allow the system administrator to set performance goals for z/OS LDAP server operations. The performance goals can be set based on the IP address of the client, distinguished name (DN) of the bound user, both the IP address and DN associated with the request, or a request matching a search pattern in the operations monitor.

The z/OS LDAP WLM support is always active and cannot be deactivated. A default service class must be configured in the WLM ISPF panels for the **LDAP** subsystem type before running the z/OS LDAP server. If a default service class is not configured in WLM, all LDAP server operations run under the discretionary or SYSOTHER profile and receive a low priority, which impacts LDAP server performance and response times. See *z/OS MVS Planning: Workload Management* for more information about WLM. See "Verifying the service class for the LDAP server" on page 604 for information about determining the service class for the LDAP servers that are running on your system.

The following WLM classification parameters are accepted on the **LDAP** subsystem type:

- **Sysplex Name (PX):** Identifies the sysplex name where the LDAP server reside. This allows an installation to establish the same performance goals for all LDAP servers running within the same sysplex.
- **Subsystem Instance (SI):** Identifies the started task name of the LDAP server. This allows an installation to establish performance goals for an individual LDAP server that is running within the sysplex.

- **Transaction Name (TN):** Identifies a WLM transaction name that has been configured in WLM. These transaction names can be used within any subsystem type in WLM. For the LDAP server, the default transaction name is GENERAL.

See "Setting up for WLM (workload management)" on page 54 for more information about setting up additional transaction names.

Installing a z/OS UNIX System Services file system for the schema backend

The LDAP server must have write access to a z/OS UNIX System Services file system to store the schema backend. The schema backend is required to run the LDAP server.

The amount of space required to store the schema backend in a z/OS UNIX System Services file system is approximately three times the size of the expected schema input LDIF files. Initially, approximately 300 kilobytes are required for storing the initial or minimum schema when the server is first started. Generally, the space required to hold the schema data is one and a half times the size of the expected input LDIF data. However, when the schema is modified a copy of the original schema file is copied, therefore, resulting in occasionally needing twice the amount of file system space.

See the following documentation for more information about how to install zFS:

- *z/OS V2R2 Program Directory*
- *z/OS Distributed File Service zFS Administration*

Optional Products

This section describes the products that may need to be installed and configured 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 is 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 60 for information about configuring the LDAP Server for accessing RACF information.	"Installing and setting up DB2 for TDBM and GDBM (DB2-based)" on page 19
SDBM backend (based on RACF)	Install RACF.	"Installing RACF for SDBM and native authentication" on page 22

If you plan to use:	You must:	See:
LDBM, CDBM, or GDBM backend (based on z/OS UNIX System Services file system)	Ensure that a z/OS UNIX System Services file system exists and has enough space to store the LDBM, CDBM, or GDBM (file based) backend and that it can be written to by the LDAP server.	"Installing a z/OS UNIX System Services file system for LDBM, GDBM (file-based), and CDBM backends" on page 22
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 22
Encryption setup	Install ICSF (optional).	"Installing ICSF for encryption, hashing, or SSL/TLS" on page 22
Kerberos authentication	Install Kerberos.	"Installing Kerberos" on page 23
Native authentication	Install a security server.	"Installing RACF for SDBM and native authentication" on page 22
Remote crypto plug-in	Install ICSF.	Remote crypto plug-in
ICTX plug-in	Install RACF.	ICTX plug-in

Installing and setting up DB2 for TDBM and GDBM (DB2-based)

This section describes how to get DATABASE 2™ (DB2), version 9 or higher, running and how to run the LDAP server using the TDBM and GDBM (DB2-based) backend. If TDBM or GDBM (DB2-based) backends are run in 64-bit mode, then DB2 version 9 with PTF UK50918 or DB2 version 10 or higher is required. For more information, see IBM Information Management Software for z/OS Solutions Information Center.

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, version 9 or later. If 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 runs. (See "Determining operational mode" on page 146 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 information about installing DB2 CLI in IBM Information Management Software for z/OS Solutions Information Center. Also, specify the user ID (for example, su*xxxx*) that should be granted database system administrator authority. This should be the ID you log on with to run the SPUFI jobs to create the DB2 tables for the LDAP server. You must find out the following information from your database administrator:
   - DB2 subsystem name. For example, DSN9.
   - 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 does not need to 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 might be necessary to have your DB2 administrator set up buffer pools, TEMP space, and TEMP data sets for additional buffer pool sizes. By default, the LDAP server DB2-based backends use buffer pool BP0. The buffer pool choice and size (4K, 32K, or other sizes) should be examined by your database system administrator to ensure that they are large enough to meet the additional needs of the LDAP server, once you have loaded data into its database. The DB2 **RUNSTATS** utility should be used once data is loaded so that DB2 queries are optimized. See Chapter 31, "Performance tuning," on page 599 for detailed information about running the DB2 **RUNSTATS** utility.

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. For more information about these parameters, see IBM Information Management Software for z/OS Solutions Information Center. 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 basic replication is being used
- if advanced replication is being used:
  - one connection for each defined replication agreement
  - one connection for each advanced replication plug-in utility operation
  - five connections for system communication for each configured TDBM backend that is used with sysplex data sharing (sysplex threads for TDBM function shifting)

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 information about setting up the DB2 CLI runtime environment in IBM Information Management Software for z/OS Solutions Information Center. 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 IBM Information Management Software for z/OS Solutions Information Center for information about bind options and syntax). Note the plan name used when editing this job, for example DSNACLI.
4. Create (Allocate) the 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 data set is used for the CLI initialization file, it must not contain sequence numbers. See the information about the DB2 CLI initialization file in IBM Information Management Software for z/OS Solutions Information Center for more information about the contents of this file. Figure 2 shows a sample file. The example in Figure 2 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=DSN9

; Example SUBSYSTEM stanza for your DB2 subsystem name
[DSN9]
;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 Resource Recovery Services attachment facility (RRSAF) to access DB2. See IBM Information Management Software for z/OS Solutions Information Center for more information about these choices.

Setting AUTOCOMMIT

To prevent data corruption, the LDAP server always uses a value of 0 for AUTOCOMMIT, regardless of the value specified in the DSNAOINI file.

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. See the following documentation for information about installing and configuring RACF:

- *z/OS V2R2 Program Directory*
- *z/OS Security Server RACF Security Administrator's Guide*
- *z/OS V2R2 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 a z/OS UNIX System Services file system for LDBM, GDBM (file-based), and CDBM backends

The LDAP server must have write access to a z/OS UNIX System Services file system to store each LDBM, GDBM (file-based), and CDBM backend that is configured.

See "Setting up for LDBM" on page 61, "Setting up for GDBM" on page 64, and "Setting up for CDBM" on page 62 for the amount of space that is needed to store the backend.

See the following documentation for more information about how to install zFS:

- *z/OS V2R2 Program Directory*
- *z/OS Distributed File Service zFS Administration*

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 LNKLST to make their libraries available. See *z/OS Cryptographic Services System SSL Programming* for more information regarding SSL/TLS. Also, see "Setting up for SSL/TLS" on page 68 for details on configuring and using SSL/TLS with your LDAP server.

Installing ICSF for encryption, hashing, or SSL/TLS

The z/OS LDAP server supports AES, crypt, DES, MD5, SHA, Salted SHA (SSHA), SHA-2, and Salted SHA-2 algorithms for encryption or hashing of **userPassword** and **ibm-slapdAdminPw** attribute values in the TDBM, LDBM, and CDBM backends. The **secretKey**, **replicaCredentials**, **ibm-replicaKeyPwd**, and **ibm-slapdMasterPw** attribute values in the TDBM, LDBM, and CDBM backends can be encrypted in either AES or DES.

The SHA-2 and Salted SHA-2 hashing algorithms consists of the following methods: SHA224, SSHA224 (Salted SHA224), SHA256, SSHA256 (Salted SHA256), SHA384, SSHA384 (Salted SHA384), SHA512, and SSHA512 (Salted SHA512). If the LDAP server is configured to use any SHA-2 or Salted SHA-2 hashing method, ICSF must be installed because the LDAP server uses the ICSF PKCS #11 omnipresent token to perform the SHA-2 and Salted SHA-2 hashing. The user ID that runs the LDAP server must have READ access to the CSFPOWH profile in the CSFSERV class to perform the SHA-2 or Salted SHA-2 hashing. See "Additional setup for using SHA-2 or Salted SHA-2 hashing" on page 51 for more information.

AES or DES keys can be stored in either an LDAPKEYS data set or in ICSF. If ICSF is not used by the z/OS LDAP server, the encryption is done either by software or hardware assists (if it is available) depending upon the encryption method selected. For more information about the LDAPKEYS data set and the other encryption or hashing methods in the z/OS LDAP server, see "Configuring for encryption or hashing" on page 77.

If using ICSF to perform AES or DES encryption, the processor must have hardware cryptographic support. All new processors have hardware cryptographic support, while some older processors optionally provide this support. Other services of ICSF needed for AES or DES encryption in the z/OS 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 that is needed for AES or DES encryption of **userPasword, secretKey, replicaCredentials, ibm-replicaKeyPwd, ibm-slapdAdminPW**, and **ibm-slapdMasterPW** attribute values. The AES and DES keys that are supported by the LDAP server depends on the underlying hardware and ICSF support. For DES encryption, a single-length, double-length, or triple-length data-encrypting key (also referred to as a data key) can be stored in the CKDS. For AES encryption, a 128-bit, 192-bit, or 256-bit data-encrypting key can be stored in the CKDS. AES and DES keys can be stored in the clear or encrypted in the CKDS. See *z/OS Cryptographic Services ICSF Overview* for more information about the AES and DES keys that are supported for your hardware and ICSF level. See the information about managing cryptographic keys and using the Key Generator Utility Program in *z/OS Cryptographic Services ICSF Administrator's Guide* for instructions about how to generate and store into CKDS a data-encrypting key for AES or 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 portions of the ICSF manuals may be useful for general background information about ICSF and Cryptographic Keys.

Note: ICSF is not required to be installed when using crypt, MD5, SHA, or SSHA.

ICSF is also used by z/OS System SSL for some of its cryptographic processing. Many of the cryptographic algorithms are available through software processing that is performed by System SSL, but some are only available by using ICSF. If you are using SSL/TLS for secure communications with your LDAP server, depending on the configuration settings and cipher suites that are used, you might need ICSF also. Some of the ICSF callable services that are used by these algorithms might be protected in the CSFSERV class within RACF. You might need to permit the LDAP server user ID access to these, depending on which algorithms are selected. Any z/OS LDAP client or z/OS LDAP client application user IDs might also need similar permission. See *z/OS Cryptographic Services System SSL Programming* for more information.

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**. See *z/OS Integrated Security Services Network Authentication Service Administration* for details about setting up this file.

Chapter 4. Configuring an LDAP server using the dsconfig utility

The LDAP configuration utility, **dsconfig**, simplifies and automates the LDAP server configuration process for GDBM (DB2-based or file-based), TDBM, LDBM, EXOP, CDBM, and SDBM backends. The **dsconfig** utility also configures the remote crypto and ICTX plug-ins. The following table shows where to find specific information about the LDAP configuration utility within this topic.

Description	Topic
Overview of how the LDAP configuration utility works and information for determining if the utility is appropriate for your LDAP server configuration	"Overview of the LDAP configuration utility"
Details describing how to use the **dsconfig** utility	"dsconfig utility" on page 27
Step-by-step instructions describing how to configure an LDAP server	"Steps for configuring an LDAP server" on page 33
Configuration confirmation	"Configuration confirmation" on page 36
Advanced configuration options	"Specifying advanced configuration options with the dsconfig utility" on page 37
How to set the time zone	"Setting the time zone" on page 38

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 the output of their job and addressing any errors that might have occurred. When all JCL jobs have completed successfully, the LDAP server can be started.

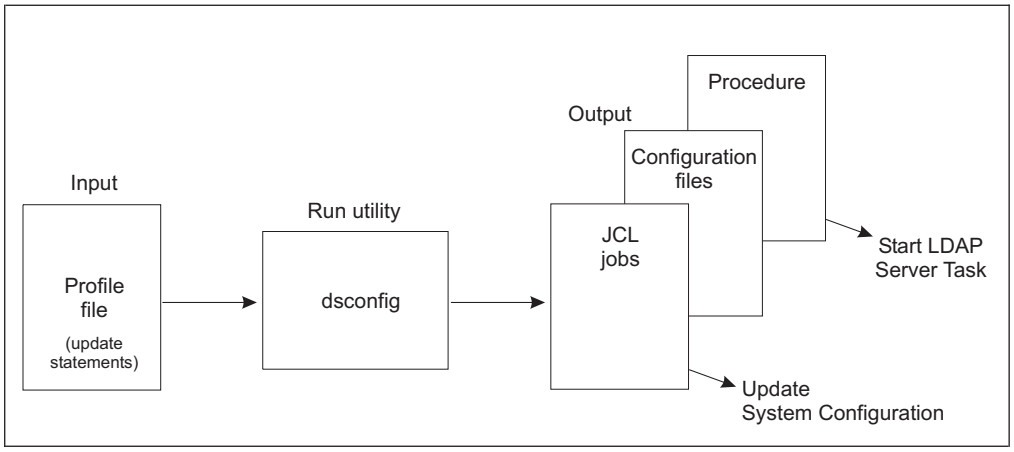


Figure 3. LDAP configuration utility overview

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 or hashing, and sysplex setup. See "Specifying advanced configuration options with the dsconfig utility" on page 37 for more information.

Capabilities

- Allows the following backends to be configured:
  - CDBM - file-based for configuration
  - EXOP - for extended operations (deprecated)
  - GDBM - DB2-based for change log
  - GDBM - file-based for change log
  - LDBM - file-based
  - SDBM - RACF-based for security
  - TDBM - DB2-based
- Allows the following plug-in extensions to be configured:
  - Remote crypto - Perform PKCS #11 or CCA services functions with ICSF
  - ICTX - Perform remote auditing and authorization checking
- 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:
  - Administrative group and roles
  - Advanced replication
  - Basic replication
  - Change logging
  - Encryption or hashing (**dsconfig** does not generate encryption key labels or data)
  - Extended operations (EXOP) backend (used for accessing Policy Directory information). This feature is deprecated.
  - Kerberos authentication
  - Native authentication
  - Password policy
  - Plug-in extensions to the server
  - Referrals
  - Secure Sockets Layer (SSL) or Transport Layer Security (TLS) (**dsconfig** does not generate certificates or passwords)

Restrictions

- Assumes that RACF is the security server in use. However, if RACF is not the security server in use, **dsconfig** could still be used. The resulting RACF JCL job must be converted to properly update the security server in use.
- Does not allow configuring multiple backends of the same database type.
- 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.
- Any manual updates to the output that the utility produces is lost if you run the utility again with the same output data set.
- The **database** *name* value can be manually updated. Edit and update the value in the LDAP server configuration file that is generated by **dsconfig**.

If you cannot use the **dsconfig** utility because of one or more of these restrictions, see Chapter 5, "Configuring an LDAP server without the dsconfig utility," on page 39 for information about alternate methods you can use to configure your LDAP server.

Running the dsconfig utility

In order to run the **dsconfig** utility in the shell, some environment variables must be set properly, as follows:

- **STEPLIB** - If SYS1.SIEALNKE is not in LNKLST, a **STEPLIB** is required before running the **dsconfig** utility. Issue:

  ```
  export STEPLIB=SYS1.SIEALNKE:$STEPLIB
  ```

- **PATH** - Ensure that **/usr/lpp/ldap/sbin** is added to the **PATH** environment variable. Issue:

  ```
  export PATH=/usr/lpp/ldap/sbin:$PATH
  ```

- **NLSPATH** - Ensure that **/usr/lpp/ldap/lib/nls/msg/%L/%N** is added to the **NLSPATH** environment variable. Issue:

  ```
  export NLSPATH=/usr/lpp/ldap/lib/nls/msg/%L/%N:$NLSPATH
  ```

- **LANG** - Ensure that the **LANG** environment variable is set properly. Issue:

  ```
  export LANG=En_US.IBM-1047
  ```

dsconfig utility

Purpose

dsconfig takes as input a profile containing a set of options that, when updated, are used to create a set of JCL members, configuration files, and the LDAP server start-up procedure. This output set configures an LDAP server and sets up the z/OS system to run the server. When the JCL jobs are submitted and are successful, the LDAP server can be started.

Format

dsconfig -i *profile_file* [**-s** *ds_file*] [**-a yes|no**] [**-d** *debug_level*] [**-?**]

Parameters

-?	Specifies the usage.

-a yes|no	Specifies a preemptive answer for the **dsconfig** prompted question to overwrite an existing output data set, from a previous execution of **dsconfig**.
-d *debug_level*	Specifies the debug level of tracing for **dsconfig**. The debug level is specified in the same fashion as the debug level for the LDAP server, as described in Table 18 on page 173. The default is no debug messages.
-i *profile_file*	Specifies the path and file name of the input profile file. This file contains the base set of configuration options for the LDAP server and options necessary for the JCL output members.
-s *ds_file*	Specifies the path and file name of the output server configuration file. **Note:** The LDAP configuration file generated with this option can be used with the method referred to at Chapter 5, "Configuring an LDAP server without the dsconfig utility," on page 39. When this option is specified, no output data set members are generated.

Examples

To execute **dsconfig** in its most typical usage, where `ds.profile` is the input profile file and in the current directory, enter:

```
dsconfig -i ds.profile
```

This example looks for the profile file in the `/home/u` directory and only generates a configuration file in the `/home/u` directory.

```
dsconfig -i /home/u/ds.profile -s /home/u/ds.conf
```

To enable error and trace debugging and preemptively skip the "overwrite data set" question, enter:

```
dsconfig -i ds.profile -d error+trace -a yes
```

Input file description

The input file, **ds.profile**, must be modified before executing the LDAP configuration utility, **dsconfig**. Make a copy of the shipped **ds.profile** file from **/usr/lpp/ldap/etc** to a directory where the modifications can be made.

In this file there are statements containing a keyword and value which must have the appropriate value for the target system being configured. There is a brief description given for each keyword and value in the file. Figure 4 shows a sample portion of the **ds.profile** file:

```
# LDAPUSRID <user_id>
#
# Description:
#    User ID for the LDAP server to run under.
#
# ---------------------------------------
LDAPUSRID = GLDSRV
```

Figure 4. Sample portion of ds.profile

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

Most of the statements in the **ds.profile** are required and those that are not required are labeled as optional. Some statements in the **ds.profile** have preassigned values; however, they might not be valid on the target system being configured. Values must be provided for all required statements in the **ds.profile** file.

These keywords are intentionally left blank in **ds.profile** and their values must be provided:

- ADMINDN - the distinguished name (DN) of the root administrator defined in the configuration file for this LDAP server
- PROG_SUFFIX - specifies a two character suffix for the 'PROG' output member. This member contains the APF authorizations required for the LDAP server.
- The following four JCL jobcard options must be provided:
  - APF_JOBCARD_1
  - PRGCTRL_JOBCARD_1
  - DB2_JOBCARD_1
  - RACF_JOBCARD_1

Examples and descriptions are included in the file. Change the value for OUTPUT_DATASET, the name of the data set where all configuration utility output is placed.

The **ds.profile** file embeds three other advanced input files. Information about these files can be found in "Specifying advanced configuration options with the dsconfig utility" on page 37. A common mistake, when specifying advanced configurations, is to copy and modify the advanced profiles but forget to modify the embedded path and name references for the modified advanced profiles in **ds.profile**.

Usage notes

1. The **dsconfig** utility automatically configures a CDBM backend in the following instances:
   - when the **CDBM_USEADVANCEDREPLICATION** statement is set to **on** or **off**, and the **SERVERCOMPATLEVEL** statement is set or defaults to 5 or greater, or
   - when the **USEPASSWORDPOLICY** statement is set to **on** and the **SERVERCOMPATLEVEL** statement is set or defaults to 6 or greater, or
   - when the **USEADMINROLES** statement is set to **on** and the **SERVERCOMPATLEVEL** statement is set or defaults to 7 or greater.
2. The output from **dsconfig** is written to a partitioned data set that you specify in **ds.profile**. If the data set does not exist, the utility allocates the output data set for you.

   The **dsconfig** utility prompts to confirm overwriting an existing output data set. When you specify **-a yes** on the **dsconfig** command line, the prompt is skipped when the output data set exists. When **-a no** is specified, the existing data set is not overwritten and **dsconfig** ends.

   When using the **-s** parameter, the value specified for the parameter is the path and file name of the LDAP server configuration file that is created. The **-a** parameter can also be used to avoid the prompt that confirms overwriting an existing file.

   For advanced LDAP configuration as described in "Specifying advanced configuration options with the dsconfig utility" on page 37, the **ds.profile**

contains pointers to the advanced configuration profile files. **dsconfig** expects to find the default pointers in the **ds.profile**.

3. The utility allows the configuration of an LDAP server which uses SSL/TLS. (See "Setting up for SSL/TLS" on page 68 for details.) It does not, however, automate the process of generating SSL/TLS certificates.
4. The utility allows the configuration of an LDAP server which uses encryption. It does not, however, automate the process of generating encryption keys. (See "Configuring for encryption or hashing" on page 77 for more information.)
5. Verify that the SYS1.SIEALNKE data set containing the LDAP code is in the LNKLST. If it is not in LNKLST, then STEPLIB must be used to locate this data set.

   The **dsconfig** utility does not provide an interface for adding STEPLIB statements to the started task procedure that it generates. Therefore, if an administrator wants 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 **ds.profile** file must have read access to the data sets specified in the new STEPLIB statements.
6. The APF JCL job does not work on a system using JES3. JES3 users must manually enter the following operator command in place of submitting the APF JCL job:

   - In SDSF, enter:

     ```
     /SET PROG=PROGsuffix
     ```

   - From the operator's console, enter:

     ```
     SET PROG=PROGsuffix
     ```

   The *suffix* above is specified on the PROG_SUFFIX statement in the **ds.profile** file.
7. The PRGMCTRL and RACF jobs that **dsconfig** generates require that the definitions listed below exist in RACF before submission. If the definitions do not exist, the jobs 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 **ds.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 **ds.profile** file)
      - *GLDHLQ*.** (where *GLDHLQ* appears on the GLDHLQ statement in the **ds.profile** file)
      - *GSKHLQ*.** (where *GSKHLQ* appears on the GSKHLQ statement in the **ds.profile** file)
      - *DSNHLQ*.** (where *DSNHLQ* appears on the DSNHLQ statement in the **ds.profile** file)
      - *CBCHLQ*.** (where *CBCHLQ* appears on the CBCHLQ statement in the **ds.profile** file)

- *OUTPUT_DATASET_HLQ* (where *OUTPUT_DATASET_HLQ* is the first qualifier of the data set name that appears on the OUTPUT_DATASET statement in the **ds.profile** file)

Note: The server operates 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 **ds.profile** file has read permission on all required data sets.

8. Administrators with the appropriate authorizations must submit the JCL jobs generated by **dsconfig** on the target system.
9. You might receive an error when running **dsconfig** from a **rlogin** session. This error return code of 12 can be ignored when running under the **rlogin** environment. The error is caused when both **dsconfig** and the **rlogin** environment free the OUTPUT_DATASET. No data is lost.
10. If an error occurs when submitting and running a **dsconfig** output JCL job or, if before submission, an administrator considers a value within the JCL job unsatisfactory, the administrator does not modify the JCL job directly. Instead, the administrator updates the appropriate profile files and perform all the steps that are outlined in "Steps for configuring an LDAP server" on page 33 again.

    To help determine the statements within a profile file that the administrator might have to update, at the top of every file that is generated by **dsconfig** there is a listing of statements that **dsconfig** used to create the output file. The administrator can use this listing to determine the exact statement within a profile file that is updated. Note when resubmitting all the JCL jobs **dsconfig** creates, many times JCL jobs for other components might have errors because of the duplication of a previous update. These messages can be ignored.
11. If the value of a statement requires a length greater than 65 within the generated DSCONFIG member, the LDAP Administrator can move the DSCONFIG 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.
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. Make all updates through the input files, running the utility again to re-create the jobs. Otherwise, if the generated JCL jobs are manually updated, those updates are lost if the utility is run again using the same output data set.
14. Be sure to use a different output data set than is currently being used by other LDAP servers.
15. The DBCLI, DSNAOINI and TDBSPUFI output members are created when the DB2-based TDBM backend is being configured.
16. The DBCLI, DSNAOINI and GDBSPUFI output members are created when the DB2-based GDBM backend is being configured.
17. The output set of JCL jobs, configuration files, and LDAP server start procedure are rewritten whenever **dsconfig** is rerun. This only happens if the OUTPUT_DATASET option in the *profile_file* is not changed. A common mistake is made when an LDAP administrator makes changes to the output of **dsconfig**, particularly the configuration files, and then at some point later, reruns **dsconfig** and writes over those administrative changes.

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.

Table 2. LDAP configuration utility roles and responsibilities

Role	Responsibility	Input file name/type	Output members
System Administrator (or System Programmer)	APF authorization	**ds.profile** (main)	• APF • PROG*suffix* (*suffix* is specified on the PROG_SUFFIX statement in **ds.profile**)
Database Administrator	DB2, CLI	**ds.db2.profile** (advanced)	• TDBSPUFI • GDBSPUFI • DBCLI • DSNAOINI
LDAP Administrator	LDAP server, Kerberos authentication, native authentication	**ds.slapd.profile** (advanced)	• *user_id* procedure (user ID is specified on the LDAPUSRID statement in **ds.profile**) • DSENVVAR • DSCONFIG
Security Administrator	RACF, SSL/TLS, password encryption or hashing	**ds.racf.profile** **ds.slapd.profile** (both files are advanced)	• RACF • PRGMCTRL

Figure 5 on page 33 is a graphical representation showing the administrative roles, input files, and output members for **dsconfig**.

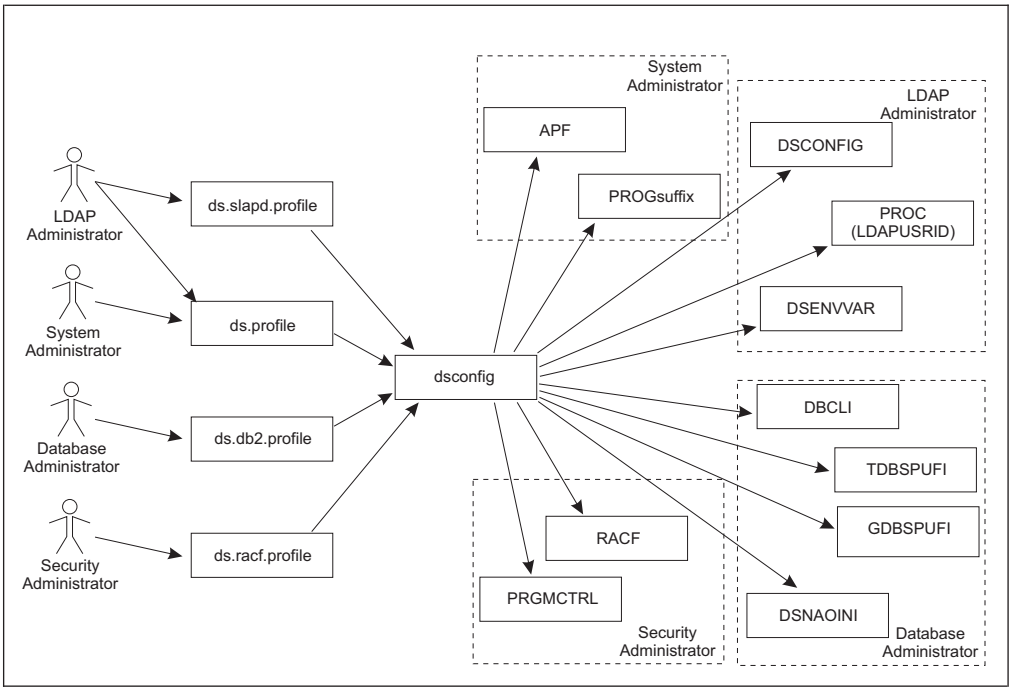


Figure 5. LDAP configuration utility roles and responsibilities

Steps for configuring an LDAP server

To configure with the configuration utility:

1. Copy the **ds.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 must update the advanced configuration files that are mentioned in Table 2 on page 32, you must copy those files also. See "Specifying advanced configuration options with the dsconfig utility" on page 37 for more details on these files.)

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

   These **ds.profile** statements must be assigned values:
   - ADMINDN
   - PROG_SUFFIX
   - APF_JOBCARD_1
   - PRGCTRL_JOBCARD_1
   - DB2_JOBCARD_1
   - RACF_JOBCARD_1

   The JOBCARD statements must be assigned regardless of their need. They are the job cards for the JCL output that is generated by **dsconfig**. The **dsconfig** utility determines what backends are configured at the same time it processes the JOBCARD statements. Note the **dsconfig** utility uses only the appropriate job cards.

   See the **ds.profile** for all statement descriptions and examples.

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

Specify a distinguished name for ADMINDN that does not contain any of the suffixes that you are defining in the configuration. Also, specify a value for ADMINPW. This enables the root administrator defined in the configuration file to bind to the LDAP server and do the final setup of the LDAP server described in step 6 on page 35. As part of that final setup, the **adminDN** option is changed and the **adminPW** option removed from the configuration file.

2. Run the **dsconfig** utility. The utility generates a set of members in a partitioned data set, as specified on the OUTPUT_DATASET statement in the **ds.profile** file.

   The **dsconfig** utility generates:
   - APF member: A JCL job which sets APF authorizations on libraries used by the LDAP server product.
   - DBCLI member: A JCL job which binds the CLI packages to DB2 and the DSNACLI plan.
   - DSCONFIG member: The LDAP server configuration file.
   - DSENVVAR member: The LDAP server environment variables file.
   - DSNAOINI member: The DB2 DSNAOINI initialization file.
   - GDBSPUFI member: A set of DB2 SQL statements to be executed using the SPUFI tool that defines database tables for the GDBM DB2-based backend.
   - PRGMCTRL member: A JCL job which sets Program Control on libraries used by the LDAP server product.
   - PROG*suffix* member: A member needed for APF authorization.
   - RACF member: A JCL job which updates RACF to allow the LDAP server to run as a started task.
   - TDBSPUFI member: A set of DB2 SQL statements to be executed using the SPUFI tool that defines database tables for the TDBM backend.
   - A procedure member needed to start the LDAP server as a started task.

   Note: DBCLI, DSNAOINI, TDBSPUFI and GDBSPUFI are conditionally generated when either of the DB2-based backends are configured.

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 is the name of the LDAP user ID specified on the LDAPUSRID statement in the **ds.profile** file. The preassigned 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 **ds.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 DB2-based GDBM is being configured

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

         The PRGMCTRL member is only required if 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 DB2-based GDBM backend is being configured.

   Note: Update the table spaces in the GDBM database to set up for row level locking. See step 3 on page 56 in "Steps for configuring an LDAP server" on page 33 for more information.
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 preassigned 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 setup of the LDAP server.
   a. If TDBM, LDBM, or CDBM is configured, modify the LDAP server schema entry to contain the schema needed for your usage of these backends. The **schema.user.ldif** and **schema.IBM.ldif** files found in the **/usr/lpp/ldap/etc** directory might contain the schema you need. **schema.IBM.ldif** requires that you first load **schema.user.ldif**.

      Note: The distinguished name (DN) of the LDAP server schema is `cn=schema`. If the ldif file containing your schema has a DN of `cn=schema,`*`suffix`*, then update the file to change the DN to `cn=schema`.

      Use the **ldapmodify** utility 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 where 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, **ds.slapd.profile**, on the LISTEN statement. The preassigned value is 389.

      binddn Is the root administrator DN of the LDAP server. The root administrator DN is specified in the **ds.profile** file on the ADMINDN statement. This value is required and not preassigned.

      passwd Is the root administrator password of the LDAP server. The root administrator password is specified in the **ds.profile** file on the ADMINPW statement. The example 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 15, "LDAP directory schema," on page 275.

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

      ```
      ldapmodify -h myhost -p 389 -D "cn=Admin" -w secret -f /usr/lpp/ldap/etc/schema.user.ldif
      ```

      More information about **ldapmodify** can be found in *z/OS IBM Tivoli Directory Server Client Programming for z/OS*.

Multiple schemas might need to be loaded before applications that use the directory works. 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 entries for each configured TDBM and LDBM backend. Suffix entries are specified in the **ds.profile** file on the TDBM_SUFFIX and LDBM_SUFFIX statements.

   Note: If you intend to load large amounts of data in LDIF format into a TDBM backend, see "ldif2ds utility" on page 235 for instructions on using the **ldif2ds** 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 **ldif2ds**.

   Use the **ldapadd** utility to load the suffix entry.

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

   See descriptions of *ldaphost*, *ldapport*, *binddn*, and *passwd* above. *file* is a file containing the entries to be loaded in LDIF format.

   For example, if `suffix.ldif` contains the following suffix entry:

   ```
   dn: o=Your Company
   objectclass: organization
   o: Your Company
   ```

   then the suffix entry can be added by **ldapadd** as follows:

   ```
   ldapadd -h myhost -p 389 -D "cn=Admin" -w secret -f suffix.ldif
   ```

   More information about **ldapadd** can be found in *z/OS IBM Tivoli Directory Server Client Programming for z/OS*.

c. Set an appropriate ACL for controlling access to change log entries for the GDBM backend, if configured. See Chapter 28, "Change logging," on page 563 for more information.

d. Remove the **adminPW** option from the LDAP server configuration file, after initial setup of the LDAP server, See "Establishing the root administrator DN and basic replication replica server DN and passwords" on page 151 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 backend, you can use **ldif2ds** or **ldapadd**. However, if you intend to load more than 100,000 directory entries, use **ldif2ds**.

Configuration confirmation

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

Run the **ldapsearch** utility to verify the configuration.

ldapsearch -h *ldaphost*-**p** *ldapport* **-D** *binddn* **-w** *passwd* **-s base -b "" "objectclass=*"**

See descriptions of *ldaphost*, *binddn*, *passwd*, and *ldapport* above.

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 contains 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 contains both naming contexts (suffixes) listed.

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

```
ldapsearch -h myhost -p 389 -D cn=admin -w secret -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 IBM Tivoli Directory Server Client Programming for z/OS*.

Specifying advanced configuration options with the dsconfig utility

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

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

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

To modify these optional statements:

1. Copy the files you want to a local directory and update them. Each input file is modified by the appropriate administrator (see Table 2 on page 32 for LDAP configuration utility roles and responsibilities).
2. Update the **ds.profile** to correctly include those modifications. Near the end of **ds.profile**, there are three statements:

   ```
   SLAPD_PROFILE = /usr/lpp/ldap/etc/ds.slapd.profile
   DB2_PROFILE = /usr/lpp/ldap/etc/ds.db2.profile
   RACF_PROFILE = /usr/lpp/ldap/etc/ds.racf.profile
   ```

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

   ```
   SLAPD_PROFILE = /home/u/ds.slapd.profile
   DB2_PROFILE = /home/u/ds.slapd.profile
   RACF_PROFILE = /usr/lpp/ldap/etc/ds.racf.profile
   ```

Advanced configuration options might require more instructions that are not covered by the LDAP configuration utility. The following table provides references for those instructions.

Configuration option	More information
Administrative group and roles	Chapter 9, “Administrative group and roles,” on page 159
Referrals	Chapter 29, “Referrals,” on page 575
Basic replication	Chapter 25, “Basic replication,” on page 469
Advanced replication	Chapter 26, “Advanced replication,” on page 487
Password encryption or hashing	“Configuring for encryption or hashing” on page 77

Configuration option	More information
Multi-server	"Determining operational mode" on page 146
Kerberos authentication	Chapter 19, "Kerberos authentication," on page 393
Native authentication	Chapter 20, "Native authentication," on page 403
CRAM-MD5 and DIGEST-MD5 authentication	Chapter 21, "CRAM-MD5 and DIGEST-MD5 authentication," on page 413
Extended operations to access Policy Director data (deprecated)	"Setting up for Policy Director extended operations" on page 66
Entry UUID support	**serverEtherAddress** option at "serverEtherAddr mac_address" on page 127
Change logging	Chapter 28, "Change logging," on page 563
Plug-in extensions to the server	*z/OS IBM Tivoli Directory Server Plug-in Reference for z/OS*
Password policy	Chapter 18, "Password policy," on page 365
Other LDAP server options	Chapter 8, "Customizing the LDAP server configuration," on page 83

Notes:

1. If the uid specified on the LDAPUID statement in the **ds.racf.profile** file is greater than 0 and the port that is specified with the LISTEN statement in the **ds.slapd.profile** file is less than 1024, the LISTEN statements that are generated in the DSCONFIG member commentary must be added to the **PROFILE.TCPIP** data set on the target system. These LISTEN statements are in the commentary directly above the **listen** option in the generated DSCONFIG member. See *z/OS V2R2.0 Communications Server: IP Configuration Guide* for more information about the **PROFILE.TCPIP** data set.
2. When using **dsconfig** 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 19, "Kerberos authentication," on page 393 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 must edit the **ds.slapd.profile** file. See "Specifying advanced configuration options with the dsconfig utility" on page 37 for more information about updating **ds.slapd.profile**. In **ds.slapd.profile**, uncomment the TIMEZONE variable and set the value you want. For more information about time zones, see *z/OS XL C/C++ Programming Guide*.

Chapter 5. Configuring an LDAP server without the dsconfig utility

This topic lists the necessary steps involved in configuring your LDAP server if you do not use the **dsconfig** utility. It may be necessary for you to use this method instead of the **dsconfig** utility, as some LDAP configuration scenarios cannot be set up with the **dsconfig** utility. More information about the **dsconfig** utility is in Chapter 4, "Configuring an LDAP server using the dsconfig utility," on page 25.

This topic 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)
  - LDBM backends (general purpose directory, file-based)
  - CDBM backend (configuration directory, file-based)
  - GDBM backend (change log directory, DB2-based and file-based)
  - EXOP backend for accessing Policy Director data (deprecated)
  - Plug-in extensions to the server
  - Secure Sockets Layer (SSL) or Transport Layer Security (TLS)
  - Password encryption or hashing
  - 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

Task	Topic
If you are configuring an SDBM (RACF-based) backend, you must:	
Install RACF	"Installing RACF for SDBM and native authentication" on page 22
Set up the user ID and security for the LDAP server	"Setting up a user ID for your LDAP server" on page 45
Set up the LDAP server for SDBM	"Setting up for SDBM" on page 60
Configure the LDAP server	Chapter 8, "Customizing the LDAP server configuration," on page 83
Run the LDAP server and finalize setup	Chapter 10, "Running the LDAP server," on page 171
See other SDBM-specific information	Chapter 17, "Accessing RACF information," on page 327

Table 3. LDAP server configuration roadmap (continued)

Task	Topic
If you are configuring a TDBM (DB2-based) backend, you must:	
Install and set up DB2	"Installing and setting up DB2 for TDBM and GDBM (DB2-based)" on page 19
Set up the user ID and security for the LDAP server	"Setting up a user ID for your LDAP server" on page 45
Create the DB2 database and table spaces for TDBM	"Creating the DB2 database and table spaces for TDBM or GDBM" on page 55
Set up the LDAP server for TDBM	"Setting up for TDBM" on page 59
Configure the LDAP server	Chapter 8, "Customizing the LDAP server configuration," on page 83
Run the LDAP server and finalize setup (including adding schema and loading data)	Chapter 10, "Running the LDAP server," on page 171
If you are configuring an LDBM backend, you must:	
Install a z/OS UNIX System Services file system	"Installing a z/OS UNIX System Services file system for LDBM, GDBM (file-based), and CDBM backends" on page 22
Set up the user ID and security for the LDAP server	"Setting up a user ID for your LDAP server" on page 45
Set up the LDAP server for LDBM	"Setting up for LDBM" on page 61
Configure the LDAP server	Chapter 8, "Customizing the LDAP server configuration," on page 83
Run the LDAP server and finalize setup (including adding schema and loading data)	Chapter 10, "Running the LDAP server," on page 171
If you are configuring a CDBM backend, you must:	
Install a z/OS UNIX System Services file system	"Installing a z/OS UNIX System Services file system for LDBM, GDBM (file-based), and CDBM backends" on page 22
Set up the user ID and security for the LDAP server	"Setting up a user ID for your LDAP server" on page 45
Set up the LDAP server for CDBM	"Setting up for CDBM" on page 62
Configure the LDAP server	Chapter 8, "Customizing the LDAP server configuration," on page 83

Table 3. LDAP server configuration roadmap (continued)

Task	Topic
Run the LDAP server and finalize setup (including adding schema and loading data)	Chapter 10, "Running the LDAP server," on page 171
If you are configuring a GDBM (DB2-based) backend, you must:	
Install and set up DB2	"Installing and setting up DB2 for TDBM and GDBM (DB2-based)" on page 19
Set up the user ID and security for the LDAP server	"Setting up a user ID for your LDAP server" on page 45
Create the DB2 database and table spaces for GDBM	"Creating the DB2 database and table spaces for TDBM or GDBM" on page 55
Set up the LDAP server for GDBM	"Configuring file-based GDBM" on page 64
Configure the LDAP server	Chapter 8, "Customizing the LDAP server configuration," on page 83
Run the LDAP server and finalize setup (including setting access control)	Chapter 10, "Running the LDAP server," on page 171
If you are configuring a GDBM (file-based) backend, you must:	
Install a z/OS UNIX System Services file system	"Installing a z/OS UNIX System Services file system for LDBM, GDBM (file-based), and CDBM backends" on page 22
Set up the user ID and security for the LDAP server	"Setting up a user ID for your LDAP server" on page 45
Set up the LDAP server for GDBM	"Configuring file-based GDBM" on page 64
Configure the LDAP server	Chapter 8, "Customizing the LDAP server configuration," on page 83
Run the LDAP server and finalize setup (including setting access control)	Chapter 10, "Running the LDAP server," on page 171
Set up the user ID and security for the LDAP server	"Setting up a user ID for your LDAP server" on page 45
Configure the LDAP server	Chapter 8, "Customizing the LDAP server configuration," on page 83
Run the LDAP server	Chapter 10, "Running the LDAP server," on page 171
If you are configuring a plug-in extension to the LDAP server, you must:	

Table 3. LDAP server configuration roadmap (continued)

Task	Topic
Create the extension	*z/OS IBM Tivoli Directory Server Plug-in Reference for z/OS*
Set up the extension	*z/OS IBM Tivoli Directory Server Plug-in Reference for z/OS*
Configure the LDAP server	*z/OS IBM Tivoli Directory Server Plug-in Reference for z/OS* and Chapter 8, "Customizing the LDAP server configuration," on page 83
Run the LDAP server	Chapter 10, "Running the LDAP server," on page 171
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	"Installing System SSL" on page 22
Set up the LDAP server for SSL/TLS	"Setting up for SSL/TLS" on page 68
Configure the LDAP server	Chapter 8, "Customizing the LDAP server configuration," on page 83
Run the LDAP server	Chapter 10, "Running the LDAP server," on page 171
If your LDAP server is going to use encryption or hashing of attribute values, you must:	
Install ICSF if using ICSF to store DES or AES keys	"Installing ICSF for encryption, hashing, or SSL/TLS" on page 22
Set up the LDAP server for encryption or hashing	"Configuring for encryption or hashing" on page 77
Configure the LDAP server	Chapter 8, "Customizing the LDAP server configuration," on page 83
Run the LDAP server	Chapter 10, "Running the LDAP server," on page 171
If your LDAP server is going to support Kerberos Authentication, you must:	
Install and configure Kerberos	"Installing Kerberos" on page 23
Start the KDC	"Setting up for Kerberos" on page 393
Update the Kerberos segment of the LDAP server's user ID	"Setting up for Kerberos" on page 393

Table 3. LDAP server configuration roadmap (continued)

Task	Topic
Generate the LDAP server's key table file (optional)	"Setting up for Kerberos" on page 393
Configure the LDAP server	Chapter 8, "Customizing the LDAP server configuration," on page 83
Run the LDAP server	Chapter 10, "Running the LDAP server," on page 171
If your LDAP server is going to support native authentication, you must:	
Install RACF or other security server	"Installing RACF for SDBM and native authentication" on page 22
Configure the LDAP server	Chapter 8, "Customizing the LDAP server configuration," on page 83
Run the LDAP server	Chapter 10, "Running the LDAP server," on page 171

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. The same value must 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 (DB2-based) Backend	
DB2 subsystem ID	• SYSTEM(DSN) value in CLI bind JCL, DSNTIJCL (see Step 3) • MVSDEFAULTSSID and SUBSYSTEM values in CLI initialization file, DSNAOINI (see Step 4)
Plan name	• PLAN value in CLI bind JCL, DSNTIJCL (see Step 3) • PLANNAME value in CLI initialization file, DSNAOINI (see Step 4) • The *zzz* value in SQL commands to grant permissions to LDAP user (see Step 2)
Database name	• `-DB_NAME-` value in SPUFI script to create database, *GLDHLQ*.SGLDSAMP(DSTDBMDB) (see Step 2) • The *yyy* value in SQL commands to grant permissions to LDAP user (see Step 5)
Database owner	• `-DB_USERID-` value in SPUFI script to create database, *GLDHLQ*.SGLDSAMP(DSTDBMDB) (see Step 2) • The **dbuserid** value in LDAP server configuration file, **ds.conf**) •
CLI initialization file name	• The **dsnaoini** value in LDAP server configuration file, **ds.conf**) • DSNAOINI DD value in JCL for LDAP server and utilities

Table 4. Configuration variable interactions (continued)

Variable	Used in:
User ID running LDAP server, **ds2ldif** and **ldif2ds** utilities	• LDAPSRV value in RACF commands to create user ID (see "Requirements for a user ID that runs the LDAP server" on page 47) • The *xxx* value in SQL commands to grant permissions to LDAP user (see Step 5) • LDAPSRV value in RACF commands to create SSL/TLS key ring (see "Creating and using key databases, key rings, or PKCS #11 tokens" on page 69) • LDAPSRV value in RACF commands to create started task (see "Defining the started task for the LDAP server" on page 171)
Server name	• DATA SOURCE value in CLI initialization file, DSNAOINI (see Step 4) • The **servername** value in LDAP server configuration file, **ds.conf**)
SDBM Backend	
User ID running LDAP server or utilities	• LDAPSRV value in RACF commands to create user ID (see "Requirements for a user ID that runs the LDAP server" on page 47) • LDAPSRV value in RACF commands to create SSL/TLS key ring (see "Creating and using key databases, key rings, or PKCS #11 tokens" on page 69) • LDAPSRV value in RACF commands to create started task (see "Defining the started task for the LDAP server" on page 171)

Setting the time zone

The LDAP server uses time values that are 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/etc/ds.envvars** to the **/etc/ldap** directory. Edit **/etc/ldap/ds.envvars**, uncomment the **TZ** environment variable and set the value you want. For more information about time zones, see *z/OS XL C/C++ Programming Guide*.

When started, the LDAP server reads an environment variable file. The default file is **/etc/ldap/ds.envvars**. This default can be changed by setting the environment variable **LDAP_DS_ENVVARS_FILE** to the full path name of the environment variable file you want. LDAP server time stamps are then generated by using a UTC value and the time zone value.

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

This topic contains information about how to configure and set up the products needed by your LDAP server.

In this section, some of the examples and descriptions reflect assumptions that might 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 variables file as described in Chapter 13, "Globalization support," on page 267. The default environment variables 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.

Setting up a user ID for your LDAP server

A separate user ID should be created to run the LDAP server. The user ID that runs the LDAP server must have the following attributes:

- The user ID must have read access to the BPX.WLMSERVER profile in the FACILITY class.
- If the BPX.SERVER profile in the FACILITY class is defined, the user ID must have update access to the profile. If the administrative group and roles are defined in RACF, the BPX.SERVER profile is required.
- The user ID must have read access to the data sets or files that are defined in the started task procedure. This includes the following files:
  - LDAP server configuration file
  - LDAP server environment variables file
  - DB2 DSNAOINI file
- The user ID must have read access to the English or Japanese message catalog files that are stored in the **/usr/lib/nls/msg** directory.
- If the administrative group and roles are defined in RACF, the LDAP class must be activated.
- The user ID must belong to a group that is the owning GID or it must be the same UID as the owning UID for all z/OS UNIX System Services file-based

backend data stores (CDBM, LDBM, file-based GDBM, and schema). A z/OS UNIX System Services file-based backend data store consists of a z/OS UNIX System Services directory and the files that are created by the LDAP server in the directory.

The user ID running the z/OS LDAP server requires access to specific locations in the z/OS UNIX System Services file systems because data might be accessed or created in these directory locations.

Note: If the UID of the user ID is zero or is granted control access to the SUPERUSER.FILESYS profile in the UNIXPRIV class, the user ID already has access to all files and directories in the z/OS UNIX System Services.

- If the LDAP server configuration file (**ds.conf**) or environment variables (**ds.envvars**) file are copied to the **/etc/ldap** directory, the user ID must have read access to the **/etc/ldap** directory.
- If an LDBM, GDBM (file-based), or CDBM backend and the **databaseDirectory** configuration option is specified in the LDAP server configuration file, the user ID must be able to create the specified directory if it does not exist. If the directory exists, then the user ID must have read and write access to it.
- If the **schemaPath** configuration option is specified in the LDAP server configuration file, the user ID must be able to create the specified directory if it does not exist. If the directory exists, then the user ID must have read and write access to it.
- If the **schemaPath** configuration option is not specified or there is an LDBM, GDBM (file-based), or CDBM backend that is configured and the **databaseDirectory** configuration option is not specified, then the user ID must be able to create directories under **/var** or the **/var/ldap** directory must exist. If the **/var/ldap** directory exists, then the user ID must have read and write access to the directory.
- If the **logfile** configuration option is specified in the LDAP server configuration file and it specifies a file in the z/OS UNIX System Services file system, the directory must exist and the user ID must have read and write access to the directory.
- If the **logfileRolloverDirectory** configuration option is specified in the LDAP server configuration file and it specifies a file in the z/OS UNIX System Services file system, the directory must exist and the user ID must have read and write access to the directory.
- If the **krbKeytab** configuration option is specified in the LDAP server configuration file and it specifies a file in the z/OS UNIX System Services files system, the user ID must have read access to the file.

Note:

1. Ownership and permissions to directories and files in the z/OS UNIX System Services file system can be granted to users or groups. The LDAP server user ID can be granted permissions to files and directories based on group membership.
2. If the UID of the user ID is nonzero and is not granted control access to the SUPERUSER.FILESYS profile in the UNIXPRIV class, **chmod, chown,** or **setfacl** commands might have to be performed to grant the necessary authorization to the appropriate directories in the z/OS UNIX System Services file systems.
3. The RACF job that is generated by the **dsconfig** utility grants control access to the SUPERUSER.FILESYS profile in the UNIXPRIV class. This authorization to the profile grants the LDAP server's user ID access to all files and directories in

z/OS UNIX System Services. If you do not want this access, issue the appropriate **chmod, chown,** or **setfacl** commands so that the LDAP server's user ID has access to the directories and files mentioned above.

4. See *z/OS UNIX System Services Command Reference* for information about the **chmod, chown,** and **setfacl** commands. See *z/OS UNIX System Services Planning* for information about setting file and directory permissions or ACLs and the SUPERUSER.FILESYS profile.

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 topic use a user ID of `LDAPSRV` in the commands provided.

Note if the UID of the user ID running the LDAP server is not zero, all console messages that are 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 to define the user ID that runs the LDAP server (substitute appropriate values for UID and GID).

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

The following RACF commands give the LDAP server access to the Workload Manager (WLM). This is required when starting the LDAP server even if WLM is not being used to classify or prioritize work within the LDAP server.

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

The following RACF commands are entered if the `BPX.SERVER` profile is defined.

```
PERMIT BPX.SERVER CLASS(FACILITY) ID(LDAPSRV) ACCESS(UPDATE)
SETROPTS RACLIST(FACILITY) REFRESH
```

When file-based backend (LDBM, GDBM (file-based), CDBM) commits are done or the schema is modified, the LDAP server creates a new version of necessary files, and updates the permissions and ownership of the files to maintain the original permissions and ownership. If the LDAP server user ID has a nonzero UID value, more permissions might be needed to allow access to files it uses and for it to perform **chmod** and **chown** commands on the schema or backend files and directories. The examples below can be used, but should be evaluated and tailored to comply with your own security policies:

- Grant control access to the SUPERUSER.FILESYS profile in the UNIXPRIV class. You might not want control access to the SUPERUSER.FILESYS profile because it grants the LDAP server user ID access to all files and directories in z/OS UNIX System Services. See *z/OS UNIX System Services Planning* for more information. Issue these RACF commands to grant control access to SUPERUSER.FILESYS:

  ```
  RDEFINE UNIXPRIV SUPERUSER.FILESYS UACC(NONE)
  PERMIT SUPERUSER.FILESYS CLASS(UNIXPRIV) ID(LDAPSRV) ACCESS(CONTROL)
  SETROPTS CLASSACT(UNIXPRIV)
  SETROPTS RACLIST(UNIXPRIV) REFRESH
  ```

- Grant read access to the SUPERUSER.FILESYS.CHOWN and SUPERUSER.FILESYS.CHANGEPERMS profiles in the UNIXPRIV class by issuing these RACF commands:

```
RDEFINE UNIXPRIV SUPERUSER.FILESYS.CHOWN UACC(NONE)
PERMIT SUPERUSER.FILESYS.CHOWN CLASS(UNIXPRIV) ID(LDAPSRV) ACCESS(READ)
RDEFINE UNIXPRIV SUPERUSER.FILESYS.CHANGEPERMS UACC(NONE)
PERMIT SUPERUSER.FILESYS.CHANGEPERMS CLASS(UNIXPRIV) ID(LDAPSRV) ACCESS(READ)
SETROPTS CLASSACT(UNIXPRIV)
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

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

Additional setup for user ID that runs the LDAP server

The user ID that runs the LDAP server must be in the group that owns the backend directory and files, or it must own the backend directory and files. If not, message GLD1342E is issued and

Indicates the UID and GIDs for the LDAP server user ID.

Indicates which file or directory it does not own.

Indicates the UID and GID of the file or directory.

For example:

```
GLD1342E Unwilling to open file or directory '/var/ldap/schema':
File or directory UID 8, UID of program 0, GID of file or
directory 8, GIDs of program (10, 0, 1, 110011).
```

In the message text:

```
- The file or directory is "/var/ldap/schema". An "ls -n" of
this path shows that it is a directory.
- The owning UID of the directory is 8.
- The owning GID of the directory is 8.
- The UID of the program is 0. Therefore, it does not own the
directory.
- The GIDs of the program are 10, 0, 1 and 110011. Therefore, it
is not in a group that owns the directory.
```

If the server has other file-based backends such as CDBM, LDBM, or file-based GDBM, then an "`ls -n`" of the backend directory (as specified in the **databaseDirectory** option in the LDAP server configuration file) shows the UID and GID of the files and directories. All of the backend files and directories must be owned by the user ID that runs the LDAP server, or be owned by one of the user ID's groups (see “Requirements for a user ID that runs the LDAP server” on page 47 for the example with the group "LDAPGRP").

The z/OS UNIX System Services chown command can be used to change the owner of the file or directory. The z/OS UNIX System Services chgrp command can be used to change the owning group of the file or directory.

Note: When the LDAP server creates the files, the owning group is inherited from the parent directory.

These requirements also apply to the **ds2ldif** and **ldif2ds** utilities. The **ds2ldif** utility accesses the schema directory and file, and the directory and files for the backend that is being unloaded. The **ldif2ds** utility accesses the schema directory and file, and the CDBM backend directory and files. Therefore, the user ID that runs either utility must be in the group that owns these directories and the files within these directories.

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 runs 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 or password phrase 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 you create KEYSMSTR class profile LDAP.BINDPW.KEY 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 hexadecimal 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 about creating and using profiles.

Additional setup for sysplex

If you plan to run the LDAP server in a sysplex group, the user ID that runs the LDAP server must have READ access to the GLD.XCF.GROUP.*group_name* resource in the FACILITY class, where *group_name* is the value of the **serverSysplexGroup** option in the LDAP server configuration file. For example, if the **serverSysplexGroup** value is LDAP, then issue the following RACF commands:

```
RDEFINE FACILITY GLD.XCF.GROUP.LDAP UACC(NONE)
PERMIT GLD.XCF.GROUP.LDAP CLASS(FACILITY) ID(LDAPSRV) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Defining the Kerberos identity

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

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

Note *ldap_prefix* must be either `ldap` or `LDAP`. Use `ldap` unless compatibility with earlier z/OS LDAP clients is needed. Also, the *hostname* must be the primary host name for the system in DNS.

If the LDAP server is located on the same machine as the Key Distribution Center (KDC), a key table 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 when the **KRB5_SERVER_KEYTAB** environment variable in the security server configuration file (**krb5.conf**) is set to 1. This can be done by issuing the following RACF commands:

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

If the LDAP server is not running on the same machine as the Kerberos KDC server, then a key table is required. The **kadmin** command can be used to create or update the key table in a directory that can be accessed by the LDAP server. The **kadmin** command must be used to get the KDC to generate the key table when the KDC does not reside on z/OS. However, the **keytab** command can be used to add the principal to the key table if the principal's password is known. The name of the key table is then specified in the **krbKeytab** option in the LDAP server configuration file.

See *z/OS Integrated Security Services Network Authentication Service Administration*for more information about configuring the KDC.

Additional setup for generating audit records

If you plan to generate LDAP SMF 83 audit records, the following RACF commands must be entered to set up the user ID that runs the LDAP server:

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

Additional setup for using securityLabel option

If you plan on specifying **securityLabel on** in the global section of the LDAP server configuration file, the following RACF commands must be entered to set up the user ID that runs the LDAP server:

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

See *z/OS UNIX System Services Planning* for more information about setting up this profile. For other LDAP server considerations in a multilevel security environment, see *z/OS Planning for Multilevel Security and the Common Criteria*.

Additional setup when defining administrative roles in RACF

If the administrative group and roles are to be defined in RACF, the BPX.SERVER profile in the FACILITY class is required and the user ID for the LDAP server must have update access to that profile. The following RACF commands must be entered to set up the user ID that runs the LDAP server.

```
RDEFINE FACILITY BPX.SERVER UACC(NONE)
PERMIT BPX.SERVER CLASS(FACILITY) ID(LDAPSRV) ACCESS(UPDATE)
SETROPTS RACLIST(FACILITY) REFRESH
```

The LDAP class must also be activated to define administrative roles in RACF. The following RACF command activates this class:

```
SETROPTS CLASSACT(LDAP)
```

Additional setup for using SHA-2 or Salted SHA-2 hashing

If you plan on using any SHA-2 or Salted SHA-2 hashing algorithms for **userPassword** and **ibm-slapdAdminPw** attribute values, the following RACF commands must be entered to set up the user ID that runs the LDAP server. The SHA-2 and Salted SHA-2 hashing algorithms consists of the following methods: SHA224, SSHA224 (Salted SHA224), SHA256, SSHA256 (Salted SHA256), SHA384, SSHA384 (Salted SHA384), SHA512, and SSHA512 (Salted SHA512).

```
RDEFINE CSFSERV CSFOWH UACC(NONE)
PERMIT CSFOWH CLASS(CSFSERV) ID(LDAPSRV) ACCESS(READ)
SETROPTS CLASSACT(CSFSERV)
SETROPTS RACLIST(CSFSERV) REFRESH
```

Protecting the environment for the LDAP server

The data set containing the LDAP server and the data sets containing any DLLs loaded by the LDAP server must be APF-authorized. Additionally, if program control is active on your system, these same data sets must be program controlled. This includes the following data sets (substitute the appropriate high-level data set name qualifier for *DB2HLQ*):

- SYS1.SIEALNKE
- SYS1.LINKLIB
- CEE.SCEERUN
- CEE.SCEERUN2
- *DB2HLQ*.SDSNLOAD
- *DB2HLQ*.SDSNLOD2 (if running the LDAP server in 64-bit)
- SYS1.CSSLIB

The data sets can be APF-authorized by placing them in a PROGnn member in the system parameter library, or by using the SETPROG operator command. For example:

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

To determine if program control is active on your system, you can use the RACF command SETROPTS LIST and examine the ATTRIBUTES output. If NOWHEN(PROGRAM) is shown, program control is not active on your system. If WHEN(PROGRAM) is shown, program control is active on your system, and the data sets above must be program controlled. For more information about using program control, see Protecting programs in *z/OS Security Server RACF Security Administrator's Guide*.

In addition, data sets used by plug-in extensions to the LDAP server must be similarly protected. See *z/OS IBM Tivoli Directory Server Plug-in Reference for z/OS* for more information.

Chapter 7. Preparing WLM, backends, sysplex, SSL/TLS, and encryption or hashing

This topic contains information about what you must do to prepare WLM, backends, sysplex, SSL/TLS, and encryption or hashing.

To run the LDAP server optimally:	See:
Configure WLM (Workload Manager)	"Setting up for WLM (workload management)" on page 54

If you plan to use:	You must:	See:
A backend, such as LDBM, TDBM, CDBM, GDBM, SDBM, or EXOP	Copy the configuration file.	"Copying the configuration files" on page 55
The sample server to set up an LDBM backend	Use the set of example files shipped with the code.	"Creating a sample server with an LDBM backend" on page 55
TDBM or GDBM (DB2-based) backend	Create the DB2 database and table spaces by using SPUFI.	"Creating the DB2 database and table spaces for TDBM or GDBM" on page 55
TDBM backend	Set up your configuration file.	"Setting up for TDBM" on page 59
SDBM backend	Set up your configuration file.	"Setting up for SDBM" on page 60
LDBM backend	Set up your configuration file.	"Setting up for LDBM" on page 61
CDBM backend	Set up your configuration file.	"Setting up for CDBM" on page 62
GDBM backend	Set up your configuration file.	"Setting up for GDBM" on page 64
EXOP backend (deprecated)	Set up your configuration file.	"Setting up for Policy Director extended operations" on page 66
Sysplex	Enable sysplex support.	"Setting up for sysplex" on page 66
SSL/TLS	Enable SSL/TLS support.	"Setting up for SSL/TLS" on page 68
Encryption or hashing	Configure encryption or hashing.	"Configuring for encryption or hashing" on page 77

Setting up for WLM (workload management)

The z/OS LDAP server supports Workload Manager (WLM) to allow an installation to set performance goals for work within the LDAP server when compared against other work that is running on the system. The **LDAP** subsystem type is reserved in WLM to allow the system administrator to set performance goals for z/OS LDAP server operations. The performance goals can be set based on the IP address of the client, distinguished name (DN) of the bound user, both the IP address and DN associated with the request, or a request matching a search pattern in the operations monitor.

The z/OS LDAP WLM support is always active and cannot be deactivated. A default service class must be configured in the WLM ISPF panels for the **LDAP** subsystem type before running the z/OS LDAP server. If a default service class is not configured in WLM, all LDAP server operations run under the discretionary or SYSOTHER profile and receive a low priority, which impacts LDAP server performance and response times. See *z/OS MVS Planning: Workload Management* for more information about WLM. See "Verifying the service class for the LDAP server" on page 604 for information about determining the service class for the LDAP servers that are running on your system.

The **wlmExcept** configuration option can be used to specify the IP address of the client, distinguished name (DN) of bound user, or both to route those requests to any configured WLM transaction name (TN). The **wlmExcept** configuration option can be specified multiple times within the LDAP server configuration file to allow multiple client IP addresses or DNs of bound users to be associated with the same or different WLM transaction name. The order of the **wlmExcept** configuration options in the LDAP server configuration file determines the order the LDAP server uses to match incoming client requests and route them to the WLM transaction name. See the **wlmExcept** configuration option for more information.

The **WLMEXCEPT** operator modify command can be used to change the routing of incoming client requests to new or different WLM transaction names while the server is running. The **WLMEXCEPT** operator modify command can be used to associate a search pattern in the **cn=operations,cn=monitor** entry to a WLM transaction name. See "Workload manager (WLM)" on page 602 for information about associating search patterns in the operations monitor entry to a WLM transaction name. Each time the **WLMEXCEPT** operator modify command is issued, the new mappings are added before any of the configured **wlmExcept** configuration options or previously issued **WLMEXCEPT** operator modify commands. See "LDAP server operator commands" on page 205 for more information about the **WLMEXCEPT** operator modify command.

During LDAP server initialization, a WLM enclave with a **GENERAL** transaction name is automatically created. WLM enclaves are created by the LDAP server for each configured **wlmExcept** configuration option. Based on how the WLM enclaves are configured, the LDAP server runs under the highest priority enclave in WLM.

Any WLM transaction names specified on **wlmExcept** configuration options or on the **WLMEXCEPT** modify operator commands must be mapped to a WLM service class. If a WLM transaction name is not mapped to a service class, those LDAP server operations are run under a discretionary or SYSOTHER profile and receive a low priority, which impacts LDAP server performance and response times.

See *z/OS MVS Planning: Workload Management* for more information about configuring WLM. See "Workload manager (WLM)" on page 602 for more information about using LDAP with WLM.

Copying the configuration files

The configuration files must 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 installation directory because any service to the files overwrites the modifications. Instead, modify them in **/etc/ldap**. The following commands copy the configuration files:

```
cp /usr/lpp/ldap/etc/ds.conf /etc/ldap/.
cp /usr/lpp/ldap/etc/ds.envvars /etc/ldap/.
```

Creating a sample server with an LDBM backend

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 an LDBM backend. The **ds.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.

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

Creating the DB2 database and table spaces for TDBM or GDBM

When using TDBM or DB2-based GDBM, the LDAP server DB2 database must be created by running a SPUFI (SQL Processor Using File Input) script from DB2 Interactive (DB2I). DB2I is a DB2 facility that provides for the running of SQL statements, DB2 (operator) commands, and utility invocation. For details about how to use DB2I and SPUFI, see IBM Information Management Software for z/OS Solutions Information Center. A sample DB2I SPUFI script to create the LDAP server DB2 database is provided. The same script is used for both TDBM and GDBM. To use it, do the following:

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

   The SPUFI script for creating the database, table spaces, tables, and indexes can be found in *GLDHLQ*.SGLDSAMP(DSTDBMDB). (*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 the SPUFI file, as shown in Table 5. (Table 4 on page 43 lists variables that are used in more than one file or configuration step. Be sure to specify the same values where necessary.) The SPUFI script provides specific instructions and information to help you determine the values to use in the table. "The DSTDBMDB SPUFI file" on page 657 shows an example of the file to edit and run in the SPUFI facility.

Table 5. TDBM value definitions for DSTDBMDB

Attribute script	Suggested value	Variable name in SPUFI script
Database name	GLDDB*	-DB_NAME-
Database owner	GLDSRV*	-DB_USERID-
User ID running the LDAP server	GLDSRV	-LDAP_USERID-
CLI plan name	DSNACLI	-DB_PLAN-

Table 5. TDBM value definitions for DSTDBMDB (continued)

Attribute script	Suggested value	Variable name in SPUFI script
Entry table space name	ENTRYTS	-ENTRYTS-
Buffer pool name for the LDAP entry table space	BP0	-ENTRYTS_BP0-
Long entry table space name	LENTRYTS	-LENTRYTS-
Buffer pool name for the LDAP long entry	BP0	-LENTRYTS_BP0-
Long attribute table space name	LATTRTS	-LATTRTS-
Buffer pool name for the LDAP long attribute	BP0	-LATTRTS_BP0-
Miscellaneous table space name	MISCTS	-MISCTS-
Buffer pool name for the LDAP miscellaneous attribute	BP0	-MISCTS_BP0-
Search table space name	SEARCHTS	-SEARCHTS-
Buffer pool name for the LDAP search table	BP0	-SEARCHTS_BP0-
Replica table space name	REPTS	-REPTS-
Buffer pool name for the LDAP replica attribute	BP0	-REPTS_BP0-
Descendants table space name	DESCTS	-DESCTS-
Buffer pool name for the LDAP descendants attribute	BP0	-DESCTS_BP0-
Storage group	SYSDEFLT	-SYSDEFLT-
Search column truncation size (VALUE in DIR_SEARCH)	32	-SEARCH_TRUNC_SIZE-
DN truncation size (DN_TRUNC in DIR_ENTRY)	32 for GDBM 64 for TDBM	-ENTRY_DN_TRUNC_SIZE-
Maximum size of a DN (DN in DIR_ENTRY)	512	-ENTRY_DN_SIZE-

Note: * This value must be unique for each database you are creating.

3. **Modify the script.**

   Use the values from Table 5 on page 55 to modify the script. You must have a unique database name and owner for each database you are creating.

   You should define GDBM backend table spaces to DB2 with row level locking, however, this is not required. You can do this by adding LOCKSIZE ROW to each CREATE TABLESPACE statement in the SPUFI script to create a GDBM database:

   ```
   CREATE TABLESPACE ttt IN yyy
   USING STOGROUP SYSDEFLT
   PRIQTY 14400
   SECQTY 7200
   LOCKSIZE ROW
   BUFFERPOOL BP0;
   ```

   where *ttt* is a table space name used in the GDBM SPUFI script and *yyy* is the database name used in the script.

An existing GDBM database can be updated for row level locking by running the following statements by using SPUFI (DB2 Interactive). Change the tables space names if you used different names when creating the database.

```
ALTER TABLESPACE yyy.ENTRYTS LOCKSIZE ROW;
ALTER TABLESPACE yyy.LENTRYTS LOCKSIZE ROW;
ALTER TABLESPACE yyy.LATTRTS LOCKSIZE ROW;
ALTER TABLESPACE yyy.SEARCHTS LOCKSIZE ROW;
ALTER TABLESPACE yyy.DESCTS LOCKSIZE ROW;
ALTER TABLESPACE yyy.MISCTS LOCKSIZE ROW;
ALTER TABLESPACE yyy.REPTS LOCKSIZE ROW;
```

where *yyy* is the database name used in the GDBM SPUFI script.

4. **Run the script 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 script that was copied and modified in the previous steps under a user ID with DB2 **SYSADM** authority. When the script completes running, scan the output data set to ensure that it ran successfully.

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

   In order to run the LDAP server, **ds2ldif**, and **ldif2ds**, certain minimum DB2 resource authorizations must be granted to the user ID or user IDs that are 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, **ds2ldif**, or **ldif2ds**, *yyy* is the database name identified in the SPUFI file and *zzz* is the CLI plan name as specified in your DB2 CLI initialization file. The grant for execute only performed once, for the LDAP server. The grant for dbadm must be done for each TDBM or GDBM backend. Run the following statements using SPUFI (DB2 Interactive):

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

   These privileges might be granted by any user ID with **SYSADM** authority.

   The LDAP server, **ds2ldif**, and **ldif2ds** require **SELECT** access to SYSIBM tables in DB2. If **SELECT** access to these tables is tightly controlled in your DB2 installation, it might be necessary to grant this access to the user ID under which the LDAP server, **ds2ldif**, or **ldif2ds** runs by performing the following operations (either using SPUFI or another means of issuing SQL commands). These grants are only done once, for the LDAP server.

   ```
   grant select on sysibm.syscolumns to xxx;
   grant select on sysibm.syscoldist to xxx;
   grant select on sysibm.systables to xxx;
   grant select on sysibm.systablepart to xxx;
   grant select on sysibm.syskeys 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 fails during start with an SQL -551 return code.

Partitioning DB2 tables for TDBM

Note: Partitioning is considered only for a TDBM database. Do not partition a GDBM database.

If you are creating a large TDBM 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 15 digit decimal number between 1 and 999999999999999. To determine the maximum value to assign to each partition, calculate the maximum value's first 4 digits and concatenate it with eleven 9s (99999999999). The formula for calculating the first 4 digits is:

```
first 4 digits of partition_max_value = (10000/number of partitions) *
partition_number - 1 where partition_number starts with 1
```

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 you want to partition your directory into 10 partitions, the maximum value for each partition is:

Partition number	Partition maximum value
1	099999999999999
2	199999999999999
3	299999999999999
4	399999999999999
5	499999999999999
6	599999999999999
7	699999999999999
8	799999999999999
9	899999999999999
10	999999999999999

If you want to partition your directory into four partitions, the maximum value for each partition is:

Partition number	Partition maximum value
1	249999999999999
2	499999999999999
3	749999999999999
4	999999999999999

Setting up for TDBM

The LDAP server provides a backend to store directory information in a DB2 database. TDBM is a general-purpose backend that can store any type of directory information.

In order to configure your LDAP server to run with the TDBM 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 55).
- You must use the following lines in your **ds.conf** file:

  ```
  database tdbm GLDBTD31/GLDBTD64
  dbuserid userid
  suffix "your_suffix"
  ```

  where *userid* is the TDBM database owner and *your_suffix* is any valid DN (distinguished name). Be sure to provide a meaningful value for the suffix.

Note:

1. Multiple TDBM backends can be configured in any given LDAP server, but, they must each use a different DB2 database. The **dbuserid** value must be different than the ones used by any other TDBM backends or DB2-based GDBM backend. Similarly, the database name used when creating the DB2 tables for the TDBM backend must be unique.
2. The attributes and object classes used by TDBM depend on your usage of TDBM. It is possible that you must add schema to the LDAP server schema. See "Setting up the schema for LDBM, TDBM, and CDBM" on page 275 for more information about adding schema to the LDAP server.
3. If the TDBM backend is run in 64-bit mode, then DB2 version 9 with PTF UK50918 or DB2 version 10 or higher is required.
4. If you specify the **dsnaoini** or **serverName** configuration option for TDBM, then you must also specify the option for each TDBM backend and DB2-based GDBM backend and the value must be the same.

Copying a TDBM database

If you want to copy an existing TDBM database to a new one, you should use **ds2ldif** to unload the existing TDBM database and **ldif2ds** to load the results into the new TDBM database.

If the new TDBM database is not on the same LDAP server as the existing one, the LDAP server schema for the target LDAP server must contain all the attributes and object classes used by the TDBM entries before they can be loaded into the new

TDBM database. You can use **ds2ldif** or **ldapsearch** to unload the LDAP server schema from the source LDAP server and then load the schema LDIF file into the target LDAP server using **ldapmodify**.

See Chapter 12, "Running and using the LDAP server utilities," on page 217 for more information about **ds2ldif** and **ldif2ds**.

You can also use the DB2 copy utility to copy the existing database to a new database. The DDL specifications defined for the existing database must be used. Otherwise, the DB2 copy utility produces unreliable results.

Follow these steps to create the new DB2 database:

1. Create the new database by using the DDL specifications defined for the existing database that you are copying or backing up.
2. Copy the existing database into the new database using the DB2 copy utility.

Note: The miscellaneous table space (MISCTS) and the replica table space (REPTS) are segmented table spaces and the RESUME option must be specified when loading these table spaces from data unloaded from the existing database.

If the new database is to be used by a new or different z/OS LDAP server, the server schema might require updates for the entries being ported. The following steps should be followed for migrating the schema to the new z/OS LDAP server configuration:

1. Unload the schema from the original z/OS LDAP server by using the **ds2ldif** or **ldapsearch** utility.
2. Start the new z/OS LDAP server without the new TDBM backend specified in the server configuration file.
3. Update the schema on the new z/OS LDAP server to match the schema on the old z/OS LDAP server by using the **ldapmodify** utility and specifying the LDIF file that was unloaded in step 1.
4. Shut down the new z/OS LDAP server and add the TDBM database line to the server configuration file and restart the new z/OS LDAP server. The new z/OS LDAP server should now start with the newly copied DB2 database.

Setting up for SDBM

The LDAP server can provide remote LDAP access to the user, group, connection, and general resource profile information stored in RACF. It also supports setting RACF options that affect classes. See Chapter 17, "Accessing RACF information," on page 327 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 55).
- You must use the following lines in your **ds.conf** file:

```
database sdbm GLDBSD31/GLDBSD64
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"
```

Note:

1. Only one SDBM backend can be defined in any given LDAP server.
2. The attributes and object classes used by SDBM are always in the LDAP server schema, except for any attributes needed for RACF custom fields.
3. The **enableResources** configuration option must be specified in your **ds.conf** file if you intend to display or manage RACF resource profiles and class options. This configuration option is also required if you want to create change log entries for changes to RACF resource profiles. See "Configuration file options" on page 90 for more information.

Setting up for LDBM

The LDAP server provides a file-based backend to store directory information in a z/OS UNIX System Services file system. LDBM is a general-purpose backend that can store any type of directory information.

The amount of space that is needed to store an LDBM backend in a z/OS UNIX System Services file system is approximately four to six times the size of the expected input LDIF data. Generally, the space that is required to hold the LDBM backend data is two to three times the size of the expected input LDIF data. However, during the LDBM commit process each of the LDBM backend files is copied, therefore, resulting in occasionally needing twice the amount of file system space.

LDBM keeps its directory in storage in the LDAP address space while the LDAP server is running. See "LDBM performance considerations" on page 606 for more information about LDBM storage usage.

When the LDAP server starts for the first time with a new LDBM backend configured, the server automatically creates the directories that are specified in the **databaseDirectory** server configuration option (or in **/var/ldap/ldbm** if the configuration option is not specified). When the directories are created, the LDAP server's user ID is the owner of these directories. The permissions on these directories grant read, write, and execute access to the LDAP server's user ID. The group that the LDAP server's user ID belongs to is granted read access to the directories. As part of the LDBM backend initialization process, the server creates **LDBM-*x*.db** files (where *x* is a number such as 1, 2, 3, and so on) for each suffix in the LDBM backend section, along with an **LDBM.ckpt** file. These files are created with the LDAP server's user ID as the owner. The default permissions on these files grant read and write access to the LDAP server's user ID while the group to which the LDAP server's user ID belongs is granted read access.

If the default LDBM backend file or directory permissions or ownership are not sufficient for your needs, they can be changed manually by issuing **chmod** and **chown** commands. The LDAP server retains any manual changes in the file or directory permissions or ownership. For additional information about the **chmod** and **chown** commands, see *z/OS UNIX System Services Command Reference*.

To configure your LDAP server to run with the LDBM 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 55).
- You must use the following lines in your **ds.conf** file:

  ```
  database ldbm GLDBLD31/GLDBLD64
  suffix "your_suffix"
  ```

  where *your_suffix* is any valid DN (distinguished name). Be sure to provide a meaningful value for the suffix.

Note:

1. Multiple LDBM backends can be configured in an LDAP server, but each must use a different file directory for storing its entries. If you are configuring multiple LDBM backends or do not want an LDBM backend to store its entries in the default file directory, then add the **databaseDirectory** option to the LDBM section of the configuration file. The **databaseDirectory** value must be different than the ones used by any other LDBM backends, file-based GDBM backend, or CDBM backend. The LDAP server must have read/write access to the file directory. See "Setting up a user ID for your LDAP server" on page 45 for more information.
2. The attributes and object classes that are used by LDBM depend on your usage of LDBM. It is possible that you must add schema to the LDAP server schema. See "Setting up the schema for LDBM, TDBM, and CDBM" on page 275 and Chapter 15, "LDAP directory schema," on page 275 for more information about adding schema to the LDAP server.
3. The files that are contained in the directory that is specified by the **databaseDirectory** server configuration option are for use by the LDAP server only. Do not copy, rename, or add any additional files to the directory specified as unintended consequences could occur during commit processing. If the files in the directory must be backed up, copy the entire directory to another location that is not in use by the LDAP server.

Copying an LDBM backend

If you want to copy an existing LDBM backend to a new one, use the **ds2ldif** utility to unload the existing LDBM backend. Then, use the **ldapadd** utility to load the new LDBM backend. If you want to retain the existing **ibm-entryuuid** attribute values for the entries, the LDAP server must be put into maintenance mode. See "Basic replication maintenance mode" on page 475 and "Advanced replication maintenance mode" on page 536 for more information.

If the new LDBM backend is not on the same LDAP server as the existing one, the LDAP server schema for the target LDAP server must contain all the attributes and object classes used by the LDBM entries before they can be loaded into the new LDBM backend. You can use **ds2ldif** or **ldapsearch** to unload the LDAP server schema from the source LDAP server and then load the schema LDIF file into the target LDAP server using **ldapmodify**.

Setting up for CDBM

The LDAP server provides the CDBM backend to store configuration information, for example, for advanced replication and password policy. CDBM is file-based, storing its directory information in a UNIX System Services file system.

The amount of space that is needed to store a CDBM backend in a z/OS UNIX System Services file system is approximately four to six times the size of the expected input LDIF data. Generally, the space that is required to hold the CDBM backend data is two to three times the size of the expected input LDIF data. However, during the CDBM commit process each of the CDBM backend files is copied, therefore, resulting in occasionally needing twice the amount of file system space.

CDBM keeps its directory in storage in the LDAP address space while the LDAP server is running.

When the LDAP server starts for the first time with the CDBM backend configured, the server automatically creates the directories that are specified in the **databaseDirectory** server configuration option (or in the directory that is specified by the **schemaPath** configuration option or in **/var/ldap/schema** if **schemaPath** is not specified). When the directories are created, the LDAP server's user ID is the owner of these directories. The permissions on these directories grant read, write, and execute access to the LDAP server's user ID. The group that the LDAP server's user ID belongs to is granted read access to the directories. As part of the CDBM backend initialization process, the server creates **LDBM-1.db** and **LDBM-2.db** files for the **cn=configuration** and **cn=ibmpolicies** suffixes, along with an **LDBM.ckpt** file. These files are created with the LDAP server's user ID as the owner. The default permissions on these files grant read and write access to the LDAP server's user ID while the group to which the LDAP server's user ID belongs is granted read access.

If the default CDBM backend file or directory permissions or ownership are not sufficient for your needs, they can be changed manually by issuing **chmod** and **chown** commands. The LDAP server retains any manual changes in the file or directory permissions or ownership. For additional information about the **chmod** and **chown** commands, see *z/OS UNIX System Services Command Reference*.

To configure your LDAP server to run with the CDBM 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 55).
- You must use the following line to your **ds.conf** file:

```
database cdbm GLDBCD31/GLDBCD64
```

Note:

1. Only one CDBM backend can be configured in an LDAP server. If the **databaseDirectory** option is not specified in the CDBM backend section, the CDBM backend stores its entries in the directory that is specified by the **schemaPath** configuration option. If the **schemaPath** configuration option is not specified, the CDBM backend data is stored in the default **schemaPath** directory that is **/var/ldap/schema**. The **databaseDirectory** value must be different than the ones used by any other LDBM backends or file-based GDBM backend. The LDAP server must have read/write access to the file directory. See "Setting up a user ID for your LDAP server" on page 45 for more information.
2. Depending on the types of entries you intend to add to the CDBM backend, you might have to add schema to the LDAP server schema for the attributes

and objectclasses that are to be used. See "Setting up the schema for LDBM, TDBM, and CDBM" on page 275 for information about adding schema to the LDAP server.

3. The files that are contained in the directory that is specified by the **databaseDirectory** or **schemaPath** server configuration options are for use by the LDAP server only. Do not copy, rename, or add any additional files to the directory specified as unintended consequences might occur during commit processing. If the files in the directory must be backed up, copy the entire directory to another location that is not in use by the LDAP server.

Setting up for GDBM

The LDAP server can provide a change log containing information about changes to:

- RACF users, groups, user-group connections, and general resource profiles
- TDBM, LDBM, and CDBM entries
- LDAP server schema entry

GDBM can be configured to store change log entries either in a z/OS UNIX System Services file system or in DB2.

Note:

1. Only one GDBM backend can be defined in any given LDAP server.
2. The attributes and object classes used by GDBM are always in the LDAP server schema.
3. See Chapter 28, "Change logging," on page 563 for additional configuration options that can be specified.
4. 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" on page 60 and "Additional required configuration" on page 565 for more information.

Configuring file-based GDBM

The amount of space that is needed to store a GDBM (file-based) backend in a z/OS UNIX System Services file system depends on how many change log entries are going to be stored and the size of the change log entries. The number of change log entries can be controlled using the **changeLogMaxEntries** and **changeLogMaxAge** options in the GDBM section of the LDAP server configuration file. The size of a change log entry is related to the size of the LDIF when adding or modifying a TDBM, LDBM, or CDBM entry, because this LDIF is inserted into the change log entry. Generally, the space that is required to hold the GDBM backend data is:

6 X (maximum number of GDBM entries) X (largest add or modify LDIF + 1000)

This includes the extra space that is needed to copy the database files during GDBM commit processing.

When the LDAP server starts for the first time with a new GDBM (file-based) backend that is configured, the server automatically creates the directories that are specified in the **databaseDirectory** server configuration option (or in **/var/ldap/gdbm** if the configuration option is not specified). When the directories are created, the LDAP server's user ID is the owner of these directories. The permissions on these directories grant read, write, and execute access to the LDAP

server's user ID. The group that the LDAP server's user ID belongs to is granted read access to the directories. As part of the GDBM (file-based) backend initialization process, the server creates an **LDBM-1.db** file for the changelog backend, along with an **LDBM.ckpt** file. These files are created with the LDAP server's user ID as the owner. The default permissions on these files grant read and write access to the LDAP server's user ID while the group to which the LDAP server's user ID belongs is granted read access.

If the default GDBM (file-based) backend file or directory permissions or ownership are not sufficient for your needs, they can be changed manually by issuing **chmod** and **chown** commands. The LDAP server retains any manual changes in the file or directory permissions or ownership. For additional information about the **chmod** and **chown** commands, see *z/OS UNIX System Services Command Reference*.

To configure your LDAP server to run with the GDBM (file-based) 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 55).
- You must use the following line in your **ds.conf** file:

```
database gdbm GLDBGD31/GLDBGD64
```

The default file directory that is used by the file-based GDBM backend to store its entries is **/var/ldap/gdbm**. If you do not want GDBM to use that file directory, then add the **databaseDirectory** option to the GDBM section of the configuration file. The file directory must be different than the ones used by any LDBM backends or the CDBM backend. The LDAP server must have read/write access to the file directory. See "Setting up a user ID for your LDAP server" on page 45 for more information.

Note: The files that are contained in the directory that is specified by the **databaseDirectory** server configuration option are for use by the LDAP server only. Do not copy, rename, or add any additional files to the directory specified as unintended consequences might occur during commit processing. If the files in the directory must be backed up, copy the entire directory to another location that is not in use by the LDAP server.

Configuring DB2-based GDBM

In order to configure your LDAP server to run with the GDBM (DB2-based) 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 55).
- You must use the following lines in your **ds.conf** file:

```
database gdbm GLDBGD31/GLDBGD64
dbuserid userid
```

  where *userid* is the GDBM database owner.

  Note:

  1. If the GDBM (DB2-based) backend is run in 64-bit mode, then DB2 version 9 with PTF UK50918 or DB2 version 10 or higher is required.

2. The **dbuserid** value must be different than the value used by any TDBM backends. If you specify the **dsnaoini** or **serverName** configuration option for GDBM, then you must also specify the option for each TDBM backend and the value must be the same.

Setting up for Policy Director extended operations

The use of the EXOP backend and the Policy Director extended operations are deprecated.

The LDAP server provides the EXOP backend to support extended operations that retrieve Policy Directory data. See Chapter 22, "Using extended operations to access Policy Director data," on page 417 for details about using this extended operations support. The EXOP backend is not needed when using any other extended operations, such as those used in advanced replication.

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

- 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 55).
- You must use the following line in the global section of the **ds.conf** file to enable Program Call support:

  ```
  listen ldap://:pc
  ```
- You must also add the following line after the global section of the **ds.conf** file:

  ```
  database exop GLDXPD31/GLDXPD64
  ```

Setting up for sysplex

An LDAP server running in a sysplex environment supports multiple instances of the same server within a cross-system coupling facility group. Each server instance runs on a separate system but shares the same schema and optionally backend databases. This provides improved workload management when the TCP/IP sysplex distributor is used to route incoming connections to available LDAP servers within the sysplex. Each server in the same group must be identical for the backends being shared to the other servers in the group, therefore, any server in the group can be used to satisfy a request. Authentication and authorization definitions are the same for all servers in the same group so that a client authenticated by one server in the group is assumed to be authenticated to all the servers in the group.

In order to configure your LDAP server to run in a sysplex:

- 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 55). When you want the same configuration on all LDAP servers within the sysplex group, you should ensure that they physically share the same configuration file. This eliminates the possible incorrect configuration of servers that exist within the same cross system coupling facility (XCF) group.
- Add the following line to the global section of your **ds.conf** file:

  ```
  serverSysplexGroup group_name
  ```

  where *group_name* specifies the cross-system coupling facility (XCF) group. All the LDAP servers sharing the same schema and backend databases must be a

member of the same XCF group. LDAP sysplex support is activated when the **serverSysplexGroup** configuration option is specified.

- Each LDAP server in the XCF group must specify the same value for the **schemaPath** configuration option and must have read/write access to the specified directory or to **/var/ldap/schema** if the option is not specified. The schema directory used must exist within a shared z/OS UNIX System Services file system and must be accessible to all servers in the XCF group. See *z/OS UNIX System Services Planning* for information about setting up a shared z/OS UNIX System Services file system.
- For each LDBM, TDBM, CDBM, and GDBM backend that is to be shared, add the following line to the backend section of your **ds.conf** file:

  ```
  multiserver on
  ```

  Note: The **multiserver** configuration option is not supported by the SDBM backend. Sysplex support for the SDBM backend is provided by RACF and not the LDAP server. See *z/OS Security Server RACF System Programmer's Guide* for information about setting up a shared RACF database.

Each LDBM, TDBM, CDBM, and GDBM backend can either be shared within the XCF group or not. To share a backend, specify **multiserver on** in the backend section in the configuration file of each LDAP server. If **multiserver off** is specified or if the **multiserver** option is not specified, then the backend is not shared and changes to the backend are not reflected in the other servers on the sysplex, even if they contain the same suffix. If GDBM or CDBM backends are configured, then all LDBM, TDBM, CDBM, and GDBM backends must be shared or they all must be not shared. If GDBM and CDBM backends are not configured, then some LDBM and TDBM backends can be shared while others are not shared.

When an LDBM, TDBM, CDBM, or GDBM backend is shared, the *name* parameter must be specified on the **database** option for that backend, and must be the same in the configuration file for each LDAP server. The configuration file on each LDAP server that is sharing a backend must specify the same suffixes for that backend. Do not specify these suffixes for a different backend. If sharing an LDBM, CDBM, or file-based GDBM database, each LDAP server must have read/write access to the z/OS UNIX System Services directory containing the database files. When an LDBM, CDBM, or file-based GDBM backend is shared, the **databaseDirectory** configuration option must have the same value, exist within a shared z/OS UNIX System Services file system, and must be accessible to all servers in the XCF group. Each server must have read/write access to the specified directory or to **/var/ldap/ldbm** (LDBM), **/var/ldap/schema** (CDBM - defaults to **schemaPath** configuration option setting), **/var/ldap/gdbm** (GDBM) if the **databaseDirectory** configuration option is not specified. See *z/OS UNIX System Services Planning* for information about setting up a shared z/OS UNIX System Services file system.

Modification requests for the LDAP server schema or for a shared backend can be directed to any LDAP server in the sysplex group. The modified schema or directory is seen by all servers within the group. Also, when performing a persistent search of a shared backend, only one persistent search request must be made to one of the LDAP servers in the sysplex. The sysplex support results in all the LDAP servers participating in the persistent search. The exception to this is if the target of the search is a TDBM backend when the **serverCompatLevel** configuration option is less than 4. In this case, an identical persistent search

request must be issued to each LDAP server in the sysplex. See "serverCompatLevel {3 | 4 | 5 | 6 | 7| 8}" on page 125 for more information about **serverCompatLevel**.

Setting up for SSL/TLS

The LDAP server contains the ability to protect LDAP access with Secure Sockets Layer (SSL) and Transport Layer Security (TLS). 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 might 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 might shut down secure communication causing a **StopTLS** alert to be sent and the server continues communication in a non-secure mode. At a later time, the client might restart secure communication by sending another **StartTLS** extended operation followed by the handshake.

Both types of connections require that SSL/TLS 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. After the LDAP client and 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 that 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, RACF key ring, or PKCS #11 token contains the public-keys that are associated with signers of certificates. These public-keys are, in reality, contained in certificates themselves. Therefore, verifying that one certificate requires the use of a different certificate, the signers 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, RACF key rings, or PKCS #11 tokens 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 of the SSL/TLS connection. If either certificate is self-signed, then that certificate must be stored in the others key database. If the certificates are signed by some other certificate signer, then the signers certificate and any certificates that this certificate depends upon must be stored in the key databases. The key databases, RACF key rings, or PKCS #11 tokens used by the LDAP client and server must also contain the certificates that are transmitted to each other during the startup of the SSL/TLS-protected communications.

Creating and using key databases, key rings, or PKCS #11 tokens

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, RACF key ring, or PKCS #11 token to be set up before SSL/TLS protected communications can begin.

The key database is a password protected file stored in the file system. This file is created and managed using a utility program provided with System SSL called **gskkyman**. See *z/OS Cryptographic Services System SSL Programming* for more information about the **gskkyman** utility. 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
```

PKCS #11 tokens are stored and protected by ICSF. The **gskkyman** utility or the **RACDCERT** command can be used to create or modify PKCS #11 tokens. ICSF uses the CRYPTOZ SAF class to determine if the issuer of the **gskkyman** utility or the **RACDCERT** command is permitted to perform the operation against a z/OS PKCS #11 token. See *z/OS Cryptographic Services System SSL Programming* for more information about the **gskkyman** utility and *z/OS Security Server RACF Command Language Reference* for more information about the **RACDCERT** command.

The user ID associated with the LDAP server must be authorized by RACF to use the PKCS #11 token. To authorize the LDAP server, you can use the RACF commands in the following example (where *NAME* is the name of the PKCS #11 token).

```
SETROPTS CLASSACT(CRYPTOZ)
RDEFINE CRYPTOZ USER.NAME UACC(NONE)
RDEFINE CRYPTOZ SO.NAME UACC(NONE)
PERMIT USER.NAME CLASS(CRYPTOZ) ID(LDAPSRV) ACCESS(READ)
PERMIT SO.NAME CLASS(CRYPTOZ) ID(LDAPSRV) ACCESS(READ)
```

Remember to refresh RACF after doing the authorizations.

```
SETROPTS RACLIST(CRYPTOZ) 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, RACF key ring, or PKCS #11 token of the LDAP client in order for SSL/TLS protected LDAP communications to work between the client and server.

If the SSL certificate that the LDAP server is going to use is not marked as the default certificate in the key database, RACF key ring, or the PKCS #11 token then the **sslCertificate** server configuration option must be specified in order to determine which SSL certificate to use.

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 typically passed to the CA with an electronic mail message or by an HTML form which is filled out using a web browser. When the CA verifies the information for the LDAP client or server, a certificate is returned to the requester, typically by an electronic mail message. The contents of the mail message are used to define the certificate in the key database, RACF key ring, or PKCS #11 token.

Enabling SSL/TLS support

The following high-level steps are required to enable SSL/TLS support for LDAP. These steps assume that the LDAP server and z/OS Cryptographic Services System SSL are installed and configured. The data sets 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, RACF key ring, or PKCS #11 token or use its label on the **sslCertificate** option in the LDAP server configuration file.
2. Configure the LDAP server to the security options you want that are related to SSL/TLS secure communications. See “Setting up the security options for the LDAP server” on page 71 for more information, that includes:
   - Defining the acceptable SSL and TLS protocol levels.
   - Defining the acceptable cipher specifications.
   - Defining the secure sockets and bimodal sockets the server uses to listen for inbound client requests.
   - Defining the type of authentication wanted.
   - Defining the server certificate to be used, and where it is located.
3. Restart the LDAP server.

Setting up the security options for the LDAP server

Several options that are related to the LDAP server SSL/TLS secure communications can be controlled by environment variables that are used by z/OS System SSL. The accepted protocol levels, the cipher suites, and suite B profile are all configured by using environment variables.

The z/OS LDAP server does not support SSL V2 protocol, and disables it from being used. SSL V3, TLS V1.0, TLS V1.1, and TLS V1.2 protocols are supported. The z/OS System SSL defaults and environment variables control which of these supported protocols are enabled or disabled. For example, the environment variable GSK_PROTOCOL_SSLV3 can be set to "ON" to enable SSL V3 protocol, or "OFF" to disable SSL V3 protocol. The environment variable GSK_PROTOCOL_TLSV1 can be set to "ON" to enable TLS V1.0 protocol, or "OFF" to disable TLS V1.0 protocol. TLS V1.1 and TLS V1.2 protocols are disabled by default. To enable TLS V1.1 protocol, set the environment variable GSK_PROTOCOL_TLSV1_1 to "ON". Similarly, to enable TLS V1.2 protocol, set the environment variable GSK_PROTOCOL_TLSV1_2 to "ON".

The z/OS LDAP server allows the cipher suites that are used in SSL/TLS secure connections to be defined externally by using z/OS System SSL environment variables and default settings, or to be defined directly with the **sslCipherSpecs** configuration option in "bit-mask" form. The "bit-mask" form is discouraged, as this method is limited in its support and provided solely for compatibility with earlier versions before z/OS V2R1. This is described more fully, later in this section. The preferred method is to specify the **sslCipherSpecs** configuration option as "sslCipherSpecs GSK_V3_CIPHER_SPECS_EXPANDED" in your z/OS LDAP server configuration file. In this case, the SSL cipher specifications are controlled externally in the environment, and you have all the control that is afforded by z/OS System SSL environment variables and defaults. You can either use the z/OS System SSL default settings by leaving the z/OS System SSL environment variable GSK_V3_CIPHER_SPECS_EXPANDED unset, or you can set the z/OS System SSL environment variable GSK_V3_CIPHER_SPECS_EXPANDED to the list of 4-character cipher suites you want in order of preference. For example, to set the cipher suites to the following two in order:

- C028 - 256-bit AES encryption with SHA-384 message authentication and ephemeral ECDH key exchange that is signed with an RSA certificate
- C027 - 128-bit AES encryption with SHA-256 message authentication and ephemeral ECDH key exchange that is signed with an RSA certificate

Include this in the general section of your LDAP server configuration file:

```
sslCipherSpecs  GSK_V3_CIPHER_SPECS_EXPANDED
```

Include this in your LDAP server environment variable file:

```
GSK_V3_CIPHER_SPECS_EXPANDED=C028C27
```

In some cases, other z/OS System SSL environment variables might define the SSL cipher specifications, such as GSK_SUITE_B_PROFILE described below. See “Environment variables used by the LDAP server” on page 179 for more information on how to specify environment variables for the LDAP server.

If the only cipher specifications required are listed in Table 8 on page 74

To restrict the SSL/TLS secure connections that are used by the z/OS LDAP server to the Suite B Profile, you must set the GSK_SUITE_B_PROFILE environment variable to the appropriate setting. See *z/OS Cryptographic Services System SSL*

Programming for more information. If the Suite B profile is enabled, this controls the acceptable SSL/TLS protocol levels and cipher suites, overriding the other settings that are mentioned above.

The following options for SSL/TLS can be set in the LDAP server configuration file. They are described in detail in "Configuration file options" on page 90.

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

Note:

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 are deprecated by the **listen** option. For more information, see the listen option.

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* | **INADDR_ANY** | **in6addr_any**`][:`*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* | **INADDR_ANY** | **in6addr_any**`][:`*portNumber*`]`

See the **listen** option for more information.

The **sslKeyRingFile** option specifies the name of the key database, the RACF key ring, or the PKCS #11 token that is used by the LDAP server. This key database, RACF key ring, or PKCS #11 token is also used for SSL/TLS protected replication. Because the replicating server might be acting as both a replica server and an LDAP server, the replica server's certificate (or CA certificate) must be contained in the replicating server's key database file, RACF key ring, or PKCS #11 token. When using a PKCS #11 token, the **sslKeyRingFile** configuration option must be set such as this (where *NAME* is the name of the PKCS #11 token): `*TOKEN*/`*NAME*

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 LDAP server configuration file. Use of a stash file provides a method of specifying a password in a form that cannot 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 or PKCS #11 token is used instead of a key database, the **sslKeyRingFilePW** and **sslKeyRingPWStashFile** cannot be specified in the configuration file.

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

When using server authentication, by setting the **sslAuth** server configuration option to **serverAuth**, 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 servers X.509 certificate during the initial SSL handshake. If the client validates the servers 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, by setting the **sslAuth** server configuration option to **serverClientAuth**, 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. The certificate is used to establish the bind identity of the client. This identity is also affected by the **sslMapCertificate** configuration option. See "Support of certificate bind" on page 76 for more information.

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 acts as if configured for server authentication only. This is for compatibility with earlier versions of the LDAP server. In addition, System SSL can be configured to fail the SSL handshake if the client does not provide a certificate after the server requests client authentication. System SSL provides an environment variable, **GSK_CLIENT_AUTH_NOCERT_ALERT**, which indicates that a client certificate must be passed to the server to prevent the SSL handshake from failing. See *z/OS Cryptographic Services System SSL Programming* for more information.

The **sslMapCertificate** option specifies whether the server maps a certificate used in a **SASL EXTERNAL** bind to the RACF user that is associated with the certificate.

The **sslCertificate** option indicates the label of the server certificate that is to be used. This option is needed if the default certificate has not been set in the key database, RACF key ring, or PKCS #11 token or if a certificate other than the default certificate is what you want.

The **sslCipherSpecs** configuration option can be specified in "bit-mask" form to directly define the cipher specifications that are accepted from clients when using SSL/TLS secure connections. The "bit-mask" form is discouraged, as this method is limited in its support and provided solely for compatibility with earlier versions before z/OS V2R1. It supports only a portion of the cipher suites available in z/OS System SSL, contains no 4-character cipher suites, and provides no order of preference. The preferred approach is to specify "sslCipherSpecs GSK_V3_CIPHER_SPECS_EXPANDED" and use the environment variable GSK_V3_CIPHER_SPECS_EXPANDED to define the list of 4-character cipher specifications, in order of preference, as described earlier in this section.

If the cipher specifications you want are included in Table 8 on page 74, and if the order of preference matches the default order that is provided by z/OS System SSL, then the "bit-mask" form of the **sslCipherSpecs** configuration option may be used with any of the values described. See the **sslCipherSpecs** configuration option for more information.

Depending upon the level of System SSL support, the list of acceptable cipher specifications might be lowered because certain specifications might not be supported by System SSL for that level of the product. Table 8 on page 74 lists the

LDAP sslCipherSpecs mask values that are accepted by the **sslCipherSpecs** option, and the related decimal, hexadecimal, and keyword values. It also lists the System SSL cipher number.

Table 8. SSL ciphers supported by the sslCipherSpecs configuration option

Cipher	Decimal value	Hexadecimal value	SSL value	Description
ALL	4294967295	xFFFFFFFF		All cipher suites
ANY	4294967295	xFFFFFFFF		All cipher suites
DES_SHA_EXPORT	512	x00000200	09	56-bit DES encryption with SHA-1 message authentication and RSA key exchange
DH_DSS_AES_128_SHA	1048576	x00100000	30	128-bit AES encryption with SHA-1 message authentication and fixed Diffie-Hellman key exchange signed with a DSS certificate
DH_DSS_AES_256_SHA	65536	x00010000	36	256-bit AES encryption with SHA-1 message authentication and fixed Diffie-Hellman key exchange signed with a DSS certificate
DH_DSS_DES_SHA	128	x00000080	0C	56-bit DES encryption with SHA-1 message authentication and fixed Diffie-Hellman key exchange signed with a DSS certificate
DH_DSS_TRIPLE_DES_SHA	8	x00000008	0D	168-bit 3DES encryption with SHA-1 message authentication and fixed Diffie-Hellman key exchange signed with a DSS certificate
DH_RSA_AES_128_SHA	2097152	x00200000	31	128-bit AES encryption with SHA-1 message authentication and fixed Diffie-Hellman key exchange signed with an RSA certificate
DH_RSA_AES_256_SHA	131072	x00020000	37	256-bit AES encryption with SHA-1 message authentication and fixed Diffie-Hellman key exchange signed with an RSA certificate
DH_RSA_DES_SHA	64	x00000040	0F	56-bit DES encryption with SHA-1 message authentication and fixed Diffie-Hellman key exchange signed with an RSA certificate
DH_RSA_TRIPLE_DES_SHA	4	x00000004	10	168-bit 3DES encryption with SHA-1 message authentication and fixed Diffie-Hellman key exchange signed with an RSA certificate
DHE_DSS_AES_128_SHA	4194304	x00400000	32	128-bit AES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with a DSS certificate
DHE_DSS_AES_256_SHA	262144	x00040000	38	256-bit AES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with a DSS certificate
DHE_DSS_DES_SHA	32	x00000020	12	56-bit DES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with a DSS certificate

Table 8. SSL ciphers supported by the sslCipherSpecs configuration option (continued)

Cipher	Decimal value	Hexadecimal value	SSL value	Description
DHE_DSS_TRIPLE_DES_SHA	2	x00000002	13	168-bit 3DES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with a DSS certificate
DHE_RSA_AES_128_SHA	8388608	x00800000	33	128-bit AES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with an RSA certificate
DHE_RSA_AES_256_SHA	524288	x00080000	39	256-bit AES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with an RSA certificate
DHE_RSA_DES_SHA	16	x00000010	15	56-bit DES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with an RSA certificate
DHE_RSA_TRIPLE_DES_SHA	1	x00000001	16	168-bit 3DES encryption with SHA-1 message authentication and ephemeral Diffie-Hellman key exchange signed with an RSA certificate
RC2_MD5_EXPORT	4096	x00001000	06	40-bit RC2 encryption with MD5 message authentication and RSA key exchange
RC4_MD5_EXPORT	8192	x00002000	03	40-bit RC4 encryption with MD5 message authentication and RSA key exchange
RC4_MD5_US	2048	x00000800	04	128-bit RC4 encryption with MD5 message authentication and RSA key exchange
RC4_SHA_US	1024	x00000400	05	128-bit RC4 encryption with SHA-1 message authentication and RSA key exchange
RSA_AES_128_SHA	16384	x00004000	2F	128-bit AES encryption with SHA-1 message authentication and RSA key exchange
RSA_AES_256_SHA	32768	x00008000	35	256-bit AES encryption with SHA-1 message authentication and RSA key exchange
TRIPLE_DES_SHA_US	256	x00000100	0A	168-bit 3DES encryption with SHA-1 message authentication and RSA key exchange

Setting up an LDAP client

As with the LDAP server, the LDAP client that wants to use SSL/TLS protected communication needs access to a key database, RACF key ring, or PKCS #11 token. 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 client's key database, RACF key ring, or PKCS #11 token.

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, RACF key ring, or PKCS #11 token. Use whatever is provided by the CA for obtaining the CA certificate. The certificate should be obtainable in a format that is acceptable to the **gskkyman** utility 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 client's own key database, RACF key ring, or PKCS #11 token and mark it as the default.

After the key database file, RACF key ring, or PKCS #11 token is created and contains the appropriate certificates, then the LDAP client is ready to perform SSL/TLS protected communications with an LDAP server. The LDAP client 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 IBM Tivoli Directory Server Client Programming for z/OS*.

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 IBM Tivoli Directory Server Client Programming for z/OS*.

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 considers the bind DN of the client for authorization purposes to be the clients DN as transmitted in the clients certificate on the handshake. The name specified in the BIND request must match the subject name in the client certificate or must be null.

The RACF **RACDCERT** command is used to associate a certificate with a RACF user ID. If **check**, **add**, or **replace** is specified on the **sslMapCertificate** configuration option, the LDAP server determines if the client certificate used during an **EXTERNAL** bind is associated with a RACF user ID, therefore, adds that RACF user ID to the bind information kept for this client.

After an **EXTERNAL** bind, the client can access LDBM, CDBM, TDBM, and GDBM backends based on the bind DN and the groups it belongs to. If the **sslMapCertificate** option is set to keep the associated RACF user ID in the bind information, the client can perform SDBM operations to access RACF data, under the context of the RACF user ID.

See *z/OS Security Server RACF Security Administrator's Guide* for more information about associating a certificate with a RACF user ID.

Configuring for encryption or hashing

The LDAP server allows the **userPassword, secretKey, replicaCredentials, ibm-replicaKeyPwd, ibm-slapdAdminPw**, and **ibm-slapdMasterPw** attribute values to be encrypted or hashed when stored in the directory. This prevents clear data from being accessed by users, including the system administrators. An administrator may configure the server to encrypt or hash **userPassword** or **ibm-slapdAdminPw** attribute values in either a one-way hashing format or a two-way symmetric encryption format. **secretKey, replicaCredentials, ibm-replicaKeyPwd**, and **ibm-slapdMasterPw** attribute values can only be encrypted in a two-way symmetric encryption format. Besides encryption or hashing, access to data stored in the directory can be protected by the directory access control mechanism.

One-way hashing formats

The one-way hashing algorithms supported by the LDAP server are crypt, MD5, SHA, Salted SHA (SSHA), SHA-2 and Salted SHA-2. The SHA-2 and Salted SHA-2 hashing algorithms consists of the following methods: SHA224, SSHA224 (Salted SHA224), SHA256, SSHA256 (Salted SHA256), SHA384, SSHA384 (Salted SHA384), SHA512, and SSHA512 (Salted SHA512). If the LDAP server is configured to use any SHA-2 or Salted SHA-2 hashing method, the server compatibility must be 7 or greater and ICSF must be installed because the LDAP server uses the ICSF PKCS #11 omnipresent token to perform the SHA-2 and Salted SHA-2 hashing. See *z/OS Cryptographic Services ICSF Overview* for more information about ICSF. The user ID that runs the LDAP server must have READ access to the CSFPOWH profile in the CSFSERV class to perform the SHA-2 or Salted SHA-2 hashing. See "Additional setup for using SHA-2 or Salted SHA-2 hashing" on page 51 for more information.

During a simple bind, the bind password is hashed using the appropriate algorithm and compared with the stored hashed **userPassword** or **ibm-slapdAdminPw** attribute value. Assuming that a client is authorized using directory access controls to see the **userPassword** or **ibm-slapdAdminPw** attribute values on a search, the values are returned to the client in one of the following manners:

1. If the **pwSearchOutput** configuration option is set to **binary** and the **userPassword** or **ibm-slapdAdminPw** attribute value is hashed in MD5 or SHA, the LDAP server returns the encryption tag (either {MD5} or {SHA}) in UTF-8 followed by the binary hash.
2. If the **pwSearchOutput** configuration option is set to **base64** and the **userPassword** or **ibm-slapdAdminPw** attribute value is hashed in MD5 or SHA, the LDAP server returns the encryption tag (either {MD5} or {SHA}) in UTF-8 followed by the base64-encoded binary hash.
3. If the **userPassword** or **ibm-slapdAdminPw** attribute value is hashed in crypt, the LDAP server returns the encryption tag ({crypt}) in UTF-8 followed by the binary hash and sent over the wire in UTF-8. The **pwSearchOutput** configuration option has no bearing on the retrieval of crypt hashed password values.
4. If the **userPassword** or **ibm-slapdAdminPw** attribute value is hashed in SHA-2 (SHA224, SHA256, SHA384, or SHA512), the LDAP server returns the encryption tag (for example, {SHA224}) in UTF-8 followed by the

base64-encoded binary hash. The **pwSearchOutput** configuration option has no bearing on the retrieval of SHA-2 hashed password values.

5. If the **userPassword** or **ibm-slapdAdminPw** attribute value is hashed in Salted SHA (SSHA) or Salted SHA-2 (SSHA224, SSHA256, SSHA384, or SSHA512), the LDAP server returns the encryption tag (for example, {SSHA}) in UTF-8 followed by the base64-encoded binary hash and salt value. The **pwSearchOutput** configuration option has no bearing on the retrieval of Salted SHA (SSHA) or Salted SHA-2 hashed password values.

The following are examples of retrieving the same SHA hashed **userPasword** attribute value, using the **ldapsearch** client utility with the **-L** option specified. For additional information about the **ldapsearch** client utility, see *z/OS IBM Tivoli Directory Server Client Programming for z/OS*.

When the **pwSearchOutput** configuration option is set to **binary**, the SHA hashed **userPassword** value is displayed by **ldapsearch** as follows:

```
userpassword:: e1NIQX2pmT42RwaBaro+JXF4UMJsnNDYnQ==
```

The **userPassword** attribute value that is returned above contains both the UTF-8 encryption tag, {SHA}, and the binary hashed data, but they have been base64-encoded by the **ldapsearch** client utility to present the value in a printable format.

When the **pwSearchOutput** configuration option is set to **base64**, the SHA hashed **userPassword** value is displayed by **ldapsearch** as follows:

```
userpassword: {SHA}qZk+NkcGgWq6PiVxeFDCbJzQ2J0=
```

The **userPassword** attribute value that is returned above is displayed as sent to the **ldapsearch** client utility because the hashed binary data after the {SHA} encryption tag has already been base64-encoded by the LDAP server.

For applications requiring retrieval of clear text passwords or data, such as middle-tier authentication agents, the directory administrator must configure the LDAP server to perform either a two-way encryption or no encryption of user passwords or data.

Two-way encryption formats

The supported two-way encryption formats include DES and AES. Two-way encryption allows the values of the **userPassword, secretKey, replicaCredentials, ibm-replicaKeyPwd, ibm-slapdAdminPw**, or **ibm-slapdMasterPw** attributes to be 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 while installation security policies might prohibit storing clear passwords in secondary permanent storage. This option satisfies both requirements. An encrypted password can be used for password matching on a simple bind and can be decrypted for return as clear text on a search request. During simple bind, the stored encrypted **userPassword** or **ibm-slapdAdminPw** attribute values are decrypted and compared with the bind password. During a search, if the client is authorized using directory access controls to see the **userPassword, secretKey, replicaCredentials, ibm-replicaKeyPwd, ibm-slapdAdminPw**, or **ibm-slapdMasterPw** attribute values, then they are decrypted and returned as clear text.

Symmetric encryption keys

The DES and AES encryption format use symmetric encryption keys. DES uses 56-bit (single length), 112-bit (double length), or 168-bit (triple length) keys while AES uses 128-bit, 192-bit, or 256-bit keys. DES or AES keys are stored in the ICSF CKDS or in a sequential or partitioned data set referenced by the LDAPKEYS DD statement. For AES keys, the LDAPKEYS data set only allows 256-bit keys.

The ICSF KGUP utility is used to store an AES or DES key in the CKDS. See the Key Generator Utility Program in *z/OS Cryptographic Services ICSF Administrator's Guide* for instructions on how to generate and store a key in the CKDS.

The AES and DES keys supported by the LDAP server depends on the underlying hardware and ICSF support. AES and DES keys can be stored in the clear or encrypted in the CKDS. See *z/OS Cryptographic Services ICSF Overview* for more information about the AES and DES keys that are supported for your hardware and ICSF level.

DES and AES keys can be stored in a sequential or partitioned data set referenced by the LDAPKEYS DD statement. The LDAPKEYS data set should be used with caution as all AES and DES keys are stored in the clear. The data set consists of fixed-length or variable-length records with a maximum record length of 255. The records are assumed to be in the IBM-1047 code page. Comment records begin with '#' or '*' and blank records are ignored. Each record in the data set defines a single key and has the following format:

```
key-label key-part-1 key-part-2 key-part-3 key-part-4
```

The fields are separated by one or more blanks. Each key part consists of 16 hexadecimal characters representing 8 bytes of the key. A DES key requires the key label and the one, two or three key parts while an AES key requires the key label and all four key parts. In a DES key, the low-order bit in each byte is a parity bit. The parity bit must be set so that there is an odd number of 1s in each byte, but the bit is not used for encryption. Therefore, DES uses 56-bits out of each 8-byte key part for encryption. An AES key does not use parity bits, therefore, the entire key (256 bits) is used for encryption.

The LDAP server checks the LDAPKEYS data set first when looking for a key label. If the key label is not found, then the LDAP server attempts to locate an AES or DES key in the ICSF CKDS.

Configuring for user and administrator password encryption or hashing

The LDAP server allows prevention of unauthorized access to user or administrator passwords in the LDBM, TDBM, and CDBM backends. The **userPassword** and **ibm-slapdAdminPw** attribute values can be encrypted or hashed when stored in the directory, which prevents clear text passwords from being accessed by any users, including the system administrators. Use of the terms "user password" and "password" pertain to the **userPassword** attribute. Use of the term "user entry" refers to an entry in LDBM, TDBM, or CDBM that contains a **userPassword** attribute. Use of the terms "administrator password" and "password" pertain to the **ibm-slapdAdminPw** attribute. Use of the term "administrator entry" refers to an entry in LDBM, TDBM, or CDBM that contains an **ibm-slapdAdminPw** attribute.

An administrator may configure the server to encrypt or hash **userPassword** and **ibm-slapdAdminPw** attribute values in either a one-way hashing format or a two-way, symmetric, encryption format. The **pwEncryption** configuration option in an LDBM, TDBM, or CDBM section of the LDAP server configuration file specifies the encryption method that is to be used to encrypt or hash the **userPassword** and **ibm-slapdAdminPw** attribute values in that LDBM, TDBM, or CDBM backend. For more information about the password encryption or hashing types, see the **pwEncryption** option at "pwEncryption {none | crypt | MD5 | SHA | SSHA | DES:keylabel | AES:keylabel}" on page 119.

After the server is configured and started, any user or administrator passwords for new user or administrator entries or modified passwords for existing user or administrator entries are encrypted or hashed before they are stored in either the LDBM, TDBM, or CDBM backend. The encrypted or hashed passwords are tagged with the encryption algorithm name so that passwords encrypted or hashed in different formats can coexist in the directory. If a tagged encrypted or hashed **userPassword** or **ibm-slapdAdminPw** attribute value is present on an add or modify operation, the attribute value is added as it is, with no additional encryption or hashing performed on the value even if the **pwEncryption** configuration option is set to a different type of encryption or hashing.

If the **pwEncryption** configuration option in an LDBM, TDBM, or CDBM backend is changed, existing passwords remain unchanged and continue to be usable. In other words, existing user and administrator password values are not automatically converted to the new encryption method or key label.

The **db2pwden** utility is provided as a migration utility to encrypt or hash all unencrypted, AES encrypted, or DES encrypted **userPassword** attribute values in the encryption or hashing method specified by the **pwEncryption** configuration option in the LDBM, TDBM, or CDBM backend. The **db2pwden** utility does not convert encrypted or hashed **ibm-slapdAdminPw** attribute values. For example, the **db2pwden** utility allows an LDAP administrator to convert passwords from AES to DES, DES to AES, or AES to crypt. The **db2pwden** utility is similar to the LDAP client utilities, such as **ldapsearch**, in that it acts such as a client to the LDAP server and has similar command-line parameters. For more information about the **db2pwden** utility, "db2pwden utility" on page 222. For more information about the LDAP client utilities, such as **ldapsearch**, see *z/OS IBM Tivoli Directory Server Client Programming for z/OS*. The **db2pwden** utility must be run by an LDAP administrator with the appropriate authority or a user with the authority to update **userPassword** values. See Chapter 9, "Administrative group and roles," on page 159 for more information about administrative role authority.

If the **pwEncryption** configuration option is changed from AES or DES encryption to another encryption or hashing method or to a different AES or DES key label, the LDAP server must have access to the original AES or DES key label so that decryption of existing **userPassword** and **ibm-slapdAdminPw** values still occurs on bind, search, and compare operations. If you want to remove the LDAP server's access to the original AES or DES key label, it is necessary to migrate all existing AES or DES **userPassword** values to the new encryption or hashing method or new AES or DES key label by using the **db2pwden** utility. To migrate all existing AES or DES **ibm-slapdAdminPw** values to the new encryption method or new AES or DES key label, each entry that contains an **ibm-slapdAdminPw** value must be searched to obtain its value. Then, these entries must be manually modified using the **ldapmodify** utility to replace the existing value with the same value so that the new encryption or hashing method is used. After all AES or DES

encrypted passwords are converted to the new encryption or hashing method or new AES or DES key label, the LDAP servers access to the original AES or DES key label can be removed.

A simple bind succeeds if the password provided in the bind request matches any of the multiple values of the **userPassword** attribute. A simple bind succeeds with an administrator entry if the password provided in the bind request matches the single **ibm-slapdAdminPw** attribute value. Note that depending on when **userPassword** or **ibm-slapdAdminPw** values are stored in the directory, different attribute values can be encrypted or hashed using different encoding methods.

When **ldif2ds** is used to load a TDBM backend, all clear text **userPassword** and **ibm-slapdAdminPw** attribute values in new entries are encrypted or hashed by the method specified on the **pwEncryption** configuration option in the LDAP server configuration file. If there is a tagged encrypted or hashed **userPassword** or **ibm-slapdAdminPw** attribute value in an entry, the attribute value is added as it is, with no additional encryption or hashing performed on the value even if the **pwEncryption** configuration option is set to a different type of encryption or hashing.

For information about the unloading of **userPassword** and **ibm-slapdAdminPw** attribute values in the **ds2ldif** utility, see "ds2ldif utility" on page 225.

Note:

1. The z/OS LDAP server does not permit **userPassword** or **ibm-slapdAdminPw** attributes in distinguished names.
2. Some important considerations for password encryption or hashing and basic replication are described in "Data encryption or hashing and basic replication" on page 471.

   If **userPassword** or **ibm-slapdAdminPw** attribute values are replicated in an advanced replication environment, the attribute values are replicated in the clear no matter the **pwEncryption** configuration option setting. Use a secure SSL connection between the supplier and consumer servers to protect this sensitive data. See "Replication agreements" on page 498 for more information.
3. 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()** or **ldap_compare()** API that matches the first eight characters of a **userPassword** or **ibm-slapdAdminPw** attribute value hashed with the crypt algorithm in the directory matches.

Configuring for secret encryption

The LDAP server allows prevention of unauthorized access to data other than user passwords in the LDBM, TDBM, and CDBM backends. The **secretKey, replicaCredentials, ibm-replicaKeyPwd**, and **ibm-slapdMasterPw** attribute values can be encrypted when stored in the directory, which prevents clear text passwords and data from being accessed by any users, including the system administrators. Use of the term "secret encryption" refers to any entry in LDBM, TDBM, or CDBM that contains **secretKey, replicaCredentials, ibm-replicaKeyPwd**, or **ibm-slapdMasterPw** attributes.

The administrator may configure an LDBM, TDBM, or CDBM backend in the server to encrypt the secret encryption attribute values in DES or AES by specifying the **secretEncryption** option in the LDBM, TDBM, or CDBM backend section of the LDAP server configuration file. For more information about

encrypting secret encryption attribute values, see the **secretEncryption** option at "secretEncryption {none | DES:keylabel | AES:keylabel}" on page 123.

After the server is configured and started, any secret encryption attribute values for new user entries or modified secret encryption attribute values for existing user entries are encrypted before they are stored in either the LDBM, TDBM, or CDBM backend. The encrypted secret encryption attribute values are tagged with the encryption algorithm name so that values encrypted in different formats can coexist in the directory.

If the **secretEncryption** configuration option in an LDBM, TDBM, or CDBM backend is changed, existing secret encryption attribute values remain unchanged and continue to be usable. In other words, existing secret encryption attribute values are not automatically converted to the new encryption method or key label.

If the **secretEncryption** configuration option is changed from AES to DES, DES to AES, or to a different AES or DES key label, the LDAP server must have access to the original AES or DES key label so that decryption of existing values still occurs on bind, search, and compare operations. If you want to remove the LDAP server's access to the original AES or DES key label, it is necessary to migrate all existing AES or DES encrypted secret encryption values to the new AES or DES key label. This is accomplished by using **ldapsearch** to retrieve all the entries that have secret encryption values. For each entry that is returned by **ldapsearch**, use **ldapmodify** to change the secret encryption to the same value returned on **ldapsearch**. By modifying secret encryption values in this manner, the LDAP server re-encrypts the values using the new AES or DES key label that is specified on the **secretEncryption** configuration option. After the conversion of all secret encryption values is completed, the LDAP server's access to the original AES or DES key label can be removed.

When **ldif2ds** is used to load a TDBM backend, all clear text secret encryption attribute values are encrypted by the method specified on the **secretEncryption** option in the LDAP server configuration file.

For information about the unloading of secret encryption attribute values, see "ds2ldif utility" on page 225.

Note: Some important considerations for secret encryption or hashing and basic replication are described in "Data encryption or hashing and basic replication" on page 471. If secret encryption attribute values are replicated in an advanced replication environment, the attribute values are replicated in the clear no matter the **secretEncryption** configuration option setting. Use a secure SSL connection between the supplier and consumer servers to protect this sensitive data. See "Replication agreements" on page 498 for more information.

Configuring for securityLabel option

If you are configuring the **securityLabel** option, you must do one of the following:

- Ensure that the LDAP server user ID has read access to the BPX.POE resource in the FACILITY class.
- Assign the LDAP server user ID a UID of 0 when the BPX.POE resource is not defined.

For other LDAP server considerations in a multilevel security environment, see *z/OS Planning for Multilevel Security and the Common Criteria*.

Chapter 8. Customizing the LDAP server configuration

The LDAP server configuration file, **ds.conf**, contains configuration options that are read once, when the LDAP server is started. Changes to this file are not put into effect until the LDAP server is restarted. The **ds.conf** file is also used by the **ds2ldif** and **ldif2ds** utilities. The following topics contain more information about the LDAP server configuration file and options that can be specified in it.

- "Creating the ds.conf file"
- "Configuration file options" on page 90
- "Configuration considerations" on page 144
- "Determining operational mode" on page 146
- "Establishing the root administrator DN and basic replication replica server DN and passwords" on page 151
- "Example configuration scenarios" on page 154

The CDBM backend contains configuration entries whose attributes represent configuration options. These configuration entries are read when the LDAP server is started. The attribute values can be changed dynamically by an LDAP modify command while the LDAP server is running. All changes take effect immediately, without restarting the LDAP server. The CDBM backend itself is configured in the LDAP server configuration file.

- "CDBM backend configuration and policy entries" on page 139

Creating the ds.conf file

This section contains information that is necessary for creating the **ds.conf** configuration file.

Locating ds.conf

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

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

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

The default LDAP server configuration file is **/etc/ldap/ds.conf**. A different configuration file can be specified using the **-f** command-line parameter when starting the LDAP server, **ds2ldif**, and **ldif2ds** utilities.

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

Configuration file format

The **ds.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.

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.

GDBM (DB2-based) backend-specific section
Contains configuration options that apply to the DB2-based GDBM change log backend. The configuration of either type of GDBM change log backend is mutually exclusive. The LDAP server ends when a file-based GDBM backend and a DB2-based GDBM backend are both configured in the same configuration file.

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

GDBM (file-based) backend-specific section
Contains configuration options that apply to the file-based GDBM change log backend. The configuration of either type of GDBM change log backend is mutually exclusive. The LDAP server ends when a file-based GDBM backend and a DB2-based GDBM backend are both configured in the same configuration file.

CDBM backend-specific section
Contains configuration options that apply to the file-based CDBM backend.

EXOP backend-specific section (deprecated)
Contains only the **database** statement necessary for the EXOP backend.

Figure 6 on page 85 shows the general format of **ds.conf**.

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

# SDBM database definition and configuration options
database SDBM GLDBSD31/GLDBSD64
<configuration options specific to SDBM backend>

# TDBM database definition and configuration options
database TDBM GLDBTD31/GLDBTD64
<configuration options specific to TDBM backend>

# DB2-based GDBM database definition and configuration options
database GDBM GLDBGD31/GLDBGD64
<configuration options specific to DB2-based GDBM backend

# LDBM database definition and configuration options
database LDBM GLDBLD31/GLDBLD64
<configuration options specific to LDBM backend>

# File-based GDBM database definition and configuration options
database GDBM GLDBGD31/GLDBGD64
<configuration options specific to file-based GDBM backend>

# CDBM database definition and configuration options
database CDBM GLDBCD31/GLDBCD64
<configuration options specific to CDBM backend>

# EXOP database definition and configuration options (deprecated)
database EXOP GLDXPD31/GLDXPD64
```

Figure 6. General format of ds.conf

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

- The configuration file contains a global section containing options that apply to the entire LDAP server, followed by one or more database backend sections that contain options that apply to a specific backend. Each backend section begins with a **database** option and continues to the next **database** option. The global section starts at the beginning of the configuration file and ends at the first **database** option. The **sizelimit** and **timelimit** options can be either global or specific to a backend: they are global if they appear in the global section and are specific to a backend if they appear in a backend section. See the **sizelimit** configuration file option at “sizeLimit num-limit ” on page 128 and the **timeLimit** configuration file option at “timeLimit num-seconds ” on page 135 for more information.
- The configuration file must be in code page IBM-1047.
- The maximum length of a line in the configuration file is 1024 characters.
- Each configuration line consists of an option and a value separated by one or more blanks. Begin each configuration option in column one. The option is not case-sensitive. The value might not be case-sensitive depending upon the option. If an argument contains one or more blank spaces, the value is enclosed in double quotation marks (for example, `"value one"`). If a value contains a double quotation mark or a backslash character (\), the double quotation mark or backslash character is preceded by a backslash character (\).

- A line that begins with a space or tab character in column one is considered a continuation of the previous line. Everything after the space or tab character is appended to the previous line. The maximum length of the initial line plus all continuation lines is 1024 characters.
- A line that begins with a number sign (#) in column one is a comment line. Comment lines can be continued and are ignored.
- A number sign (#) can be used to add a comment to the end of a configuration line. The number sign (#) must be separated from the option value by at least one blank. Anything following the number sign (#) is ignored, including any continuation lines. A number sign (#) is not treated as the start of a comment if it occurs within a quoted value.
- Blank lines can be used to separate configuration lines.
- Options that expect a value of **on** or **off** also accepts **yes, y, no,** and **n**. The option value is not case-sensitive.
- For single-valued options that appear more than once, the last appearance in the **ds.conf** file is used.

Specifying a value for filename

For configuration file options that contain the *filename* value, the value 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 file system.

filename
: Specifies a file name in the z/OS UNIX System Services file system that is relative to the current working directory of the LDAP server. The LDAP server sets the current working directory to the value of the **HOME** environment variable or to `/etc/ldap` if the **HOME** environment variable is not defined. This format is not suggested.

//'*dataset.name***'**
: Specifies the fully qualified name of a file stored in a sequential data set.

//'*dataset.name***(***member***)'**
: Specifies the fully qualified name of a file stored in a partitioned data set.

//DD:*DDNAME*
: Specifies the DDNAME for a JCL DD statement that defines the file. The file might be in a z/OS UNIX System Services file system, a sequential data set, or a member of a partitioned data set.

Specifying a value for a distinguished name

The value for the following configuration options is a distinguished name (DN): **adminDN, krbLDAPAdmin, masterServerDN, peerServerDN, nativeAuthSubtree**, and **suffix**. Also, the **wlmExcept** configuration option optionally accepts a distinguished name (DN). Special characters (as identified in RFC 2253: *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*) used in the DN must be properly escaped using two back slashes (`\\`). Note that the double back slashes are only needed in the configuration file; in all other usages, the special characters are typically prefixed by a single backslash. See Chapter 17, "Accessing RACF information," on page 327 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. Also, see "Distinguished name syntax" on page 272 for more information about distinguished name syntax.

For example, to use a RACF user ID `#1admin` as the LDAP root administrator defined in the configuration file, the **adminDN** configuration option might look like:

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

With LDAP V3 support, UTF-8 characters can be used for textual attributes stored in the directory. It is preferred 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 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 a DN. 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 that are not within the X'00' - X'7F' range (7-bit ASCII that 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. Therefore, if you want to create the following distinguished name:

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

specify the DN 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. For `<U umlaut>`, the Unicode code point is X'00DC'. Converted to UTF-8, this character is a multi-byte sequence: X'C39C'. (See "UTF-8 support" on page 267 for conversion information.) Converted to the escaped form for input into the DN field, this character is represented as "&303&234" because X'C3' is octal 303 and X'9C' is octal 234. Therefore, the DN above is entered as:

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

If there is a case where you must enter a DN that 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".

There are several restrictions concerning the attributes used in a DN:

1. The default matching rule for an attribute used in a DN that is not contained in the LDAP server schema is **caseIgnoreMatch**. This results in this part of the DN being normalized by using uppercase in the attribute value. If the attribute is later defined in the schema and the matching rule is not **caseIgnoreMatch**, that part of the DN might now be normalized differently. Therefore, the normalized version of the DN used in an operation might not match the normalized version of the DN in the configuration file, resulting in the failure of the operation.

   In particular, if an attribute used in a suffix is not in the LDAP server schema and is later added to the schema with a different matching rule, then restart the LDAP server after the schema definition is added. Otherwise, operations using that suffix can fail.

2. Do not use an attribute with Integer syntax (1.3.6.1.4.1.1466.115.121.1.27) in a DN in the configuration file, especially in a suffix. Integer attribute values in a normalized DN have a special format that might cause problems within the LDAP server.

Configuration file checklist

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

Table 9. Configuration file options checklist

Section/topic	Options
Global	**adminDN** is required. **adminPW, allowAnonymousBinds, altServer, armName, audit, commThreads, db2StartUpRetryInterval, db2StartUpRetryLimit, db2Terminate, digestRealm, dnCacheSize, idleConnectionTimeout, include, listen, maxConnections, operationsMonitor, operationsMonitorSize, pcIdleConnectionTimeout, pcThreads, plugin, pwSearchOutput, referral, schemaPath, schemaReplaceByValue, securityLabel, sendV3stringsoverV2as, serverCompatLevel, serverEtherAddr, sizeLimit, srvStartUpError, tcpTerminate, timeLimit, validateincomingV2strings**, and **wlmExcept** are optional.
SSL/TLS	**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 when the connection is established. **sslAuth, sslCertificate, sslCipherSpecs, sslKeyRingFilePW**, **sslKeyRingPWStashFile**, and **sslMapCertificate** are optional.
Sysplex	**serverSysplexGroup** is required.
Kerberos	**supportKrb5** is required. **krbKeytab**, **krbLDAPAdmin**, and **serverKrbPrinc** are optional.
Activity logging	**logfile** is required. **logFileFilter, logFileMicroseconds, logFileMsgs, logFileOps, logFileRecordType, logFileRolloverDirectory, logFileRolloverSize, logFileRolloverTOD**, and **logFileVersion** are optional.
SDBM backend	**database** and **suffix** are required. **enableResources**, **include**, **readOnly**, **sizeLimit** and **timeLimit** are optional.
Kerberos	**krbIdentityMap** is required.

Table 9. Configuration file options checklist (continued)

Section/topic	Options
TDBM backend	**database, dbuserid,** and **suffix** are required. **aclSourceCacheSize, attrOverflowCount, attrOverflowSize, changeLoggingParticipant, dnToEidCacheSize, dsnaoini, entryCacheSize, entryOwnerCacheSize, extendedGroupSearching, filterCacheBypassLimit, filterCacheSize, include, persistentSearch, readonly, serverName, sizeLimit,** and **timeLimit** are optional.
Attribute encryption or hashing	**pwCryptCompat, pwEncryption,** and **secretEncryption** are optional.
Basic replication	Either **peerServerDN** or both **masterServer** and **masterServerDN** are required. **peerServerPW** and **masterServerPW** are optional.
Multi-server	**multiserver** is required.
Kerberos	**krbIdentityMap** is required.
Native authentication	**useNativeAuth** and **nativeAuthSubtree** are required. **nativeUpdateAllowed** is optional.
GDBM backend (DB2-based)	**database** and **dbuserid** are required. **aclSourceCacheSize, attrOverflowSize, changeLogging, changeLoggingParticipant, changeLogMaxAge, changeLogMaxEntries, dnToEidCacheSize, dsnaoini, entryCacheSize, entryOwnerCacheSize, filterCacheBypassLimit, filterCacheSize, include, persistentSearch, readonly, serverName, sizeLimit,** and **timeLimit** are optional.
Multi-server	**multiserver** is required.
LDBM backend	**database** and **suffix** are required. **attrOverflowCount, changeLoggingParticipant, commitCheckpointEntries, commitCheckpointTOD, databaseDirectory, extendedGroupSearching, fileTerminate, filterCacheBypassLimit, filterCacheSize, include, persistentSearch, readOnly, sizeLimit,** and **timeLimit** are optional.
Attribute encryption or hashing	**pwCryptCompat, pwEncryption,** and **secretEncryption** are optional.
Basic replication	Either **peerServerDN** or both **masterServer** and **masterServerDN** are required. **peerServerPW** and **masterServerPW** are optional.
Multi-server	**multiserver** is required.
Kerberos	**krbIdentityMap** is required.
Native authentication	**useNativeAuth** and **nativeAuthSubtree** are required. **nativeUpdateAllowed** is optional.

Table 9. Configuration file options checklist (continued)

Section/topic	Options
GDBM backend (file-based)	**database** is required. **changeLogging, changeLoggingParticipant, changeLogMaxAge, changeLogMaxEntries, commitCheckpointEntries, commitCheckpointTOD, databaseDirectory, fileTerminate, filterCacheBypassLimit, filterCacheSize, include, persistentSearch, readOnly, sizeLimit,** and **timeLimit** are optional.
Multi-server	**multiserver** is required.
CDBM backend	**database** is required. **attrOverflowCount, changeLoggingParticipant, commitCheckpointEntries, commitCheckpointTOD, databaseDirectory, extendedGroupSearching, fileTerminate, filterCacheBypassLimit, filterCacheSize, include, persistentSearch, readOnly, sizeLimit,** and **timeLimit** are optional.
Attribute encryption or hashing	**pwCryptCompat, pwEncryption,** and **secretEncryption** are optional.
Advanced replication	**useAdvancedReplication** is required.
Multi-server	**multiserver** is required.
Kerberos	**krbIdentityMap** is required.
Native authentication	**useNativeAuth** and **nativeAuthSubtree** are required. **nativeUpdateAllowed** is optional.
EXOP backend (deprecated)	**database** is required. **include** is optional.

Note: The **adminDN** configuration option is required and specifies the LDAP root administrator in the configuration file. The password for this LDAP root administrator can be specified in a directory entry or in the **adminPW** configuration option. See "Establishing the root administrator DN and basic replication replica server DN and passwords" on page 151 for more information. Note the use of the **adminPW** configuration option is discouraged. Instead, an existing entry in the directory is designated as the **adminDN**.

If the distinguished name specified for the **adminDN** configuration option does not exist within a configured backend suffix and the password is specified in the **adminPW** configuration option, password policy is not supported for the LDAP root administrator defined in the configuration file.

Configuration file options

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

Note: Some GDBM options can only be specified when GDBM is configured to be DB2-based and others can only be used when GDBM is file-based. See

"Configuration file checklist" on page 88 for a list of which options can be configured for each type of GDBM configuration.

aclSourceCacheSize *num-entries*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	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	LDBM	SDBM	GDBM	CDBM	EXOP
X						

The distinguished name (DN) of the root administrator for this LDAP server. Typically, this DN has unrestricted access to all entries in the directory except for entries in backends that are read-only replicas. When the LDAP server is in maintenance mode, the LDAP root administrator has unrestricted access to all entries in the directory. Select a name that is descriptive of the person that knows and administers the LDAP server. The format of the name must be in DN format that is described in Chapter 14, "Data model," on page 271. You might want the DN to have the same suffix as one of the **suffix** option values in the configuration file.

"Establishing the root administrator DN and basic replication replica server DN and passwords" on page 151 describes how to set up this root administrator DN. Additional root administrators can be defined using the administrative group and assigning the root administrator role. See Chapter 9, "Administrative group and roles," on page 159 for more information.

For information about specifying a value for a distinguished name for this option, see "Specifying a value for a distinguished name" on page 86.

adminPW *string*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

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

"Establishing the root administrator DN and basic replication replica server DN and passwords" on page 151 describes how to set up your administrator password.

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

allowAnonymousBinds {on | off}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies whether an LDAP client can perform unauthenticated operations on the LDAP server. If **off**, clients must explicitly bind to the server with a distinguished name. If **on**, a client might access the server without binding with a distinguished name and has access to data as a member of the `cn=anybody` group. See Chapter 24, "Using access control," on page 433 for more information about access control of directory data.

Default = **on**

altServer *ldap_URL*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies an equivalent server to this LDAP server. It might not be a replica, but contains the same naming contexts. There is no required format for the value, however, LDAP URL format is most commonly used and supported by LDAP clients. See "listen ldap_URL" on page 104 for a description of LDAP URL format. The option might be specified multiple times to define more than one alternate server. The alternate servers are placed in the **altServer** attribute in the root DSE and can be queried by LDAP clients to determine other servers that might be contacted in case this server is not available at some later time.

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

armName *name*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the name that the LDAP server uses when registering with the Automatic Restart Management (ARM) service. The name is 1-7 characters and can consist of letters, numbers, and the special characters '$ # @ _'. Lowercase letters are converted to uppercase. The first character might be a number. The system name is appended to form the element name. The **armName** configuration option must be specified if there are multiple instances of the LDAP server on the same system and ARM processing is enabled. See *z/OS MVS Setting Up a Sysplex* for more information about automatic restart manager.

For example, for system DCESEC4, specifying:

```
armName LDAP1
```

results in the element name LDAP1_DCESEC4.

The LDAP server registers with ARM using the element name formed from the **armName** configuration option, an element type of SYSLDAP, an element bind of CURJOB, and a termination type of ELEMTERM. See the description of the IXCARM macro in *z/OS MVS Programming: Sysplex Services Reference* for more information about these parameters and how to override them using the current ARM policy.

Default = GLDSRVR

attrOverflowCount *count*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X			X	

For TDBM, specifies the number of attribute values that are required to store the attribute values in a long attribute value table. The choice of this value allows large multi-valued attributes such as group membership lists to be stored in a separate table with its own index.

For LDBM and CDBM, specifies the number of attribute values that are required to store the attribute values in an internal indexed table, providing quicker access to the values of large multi-valued attributes such as group membership lists.

The value must be either 0 or in the range 64 to 2147483647. A value of 0 disables attribute overflow that is based on the attribute value count.

Default = 512

attrOverflowSize *num-of-bytes*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X			X		

Specifies, in bytes, the minimum size of an attribute value that is 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 maximum size of this value is 2147483647. A value of 0 disables attribute overflow that is based on attribute size.

Default = 255

audit {on | off | all,*operations* **| error,***operations* **| none,***operations***}**

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Turns LDAP auditing on or off and specifies which operations are to be audited and the associated audit level. When auditing is on, an LDAP SMF type 83 subtype 3 audit record is generated for an operation if the operation is specified on an **audit** option and the operation result matches the audit level.

This option can be specified multiple times, once to turn auditing on or off and once or more times for each audit level to specify the operations to audit for that level. Multiple operations can be specified for a level by

either putting a **+** between them on the **audit** option or by specifying multiple **audit** options with the same level.

Operations can be audited all the time or only when they fail. The following audit levels are supported:

all
An LDAP audit record is generated for the specified operations.

error
An LDAP audit record is generated for the specified operations when they fail.

none
An LDAP audit record is not generated for the specified operations.

The supported values for *operations* can be one or more of: **add, bind, compare, connect, delete, disconnect, exop, modify, modifydn, search, unbind**.

If an operation is specified in more than one level, the last level is used for the operation. If an operation is not specified in any level, the level defaults to **none** for that operation.

The LDAP server **AUDIT** operator modify command can be used to change the audit settings and to turn audit on or off while the LDAP server is running. See "LDAP server operator commands" on page 205 for more information.

Default = **off**

For example, the following **audit** options turn on auditing for modify, search, and bind failures and for all add operations. The other operations are not audited.

```
audit error,modify+search+bind
audit all,add
audit on
```

changeLogging {on | off}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	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 {on | off}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X		X	X	

Allows/disallows change logging for changes that are made to entries in this backend. When specified in GDBM, **changeLoggingParticipant** controls the logging of modifications to the LDAP server schema entry.

Note: This option does not turn on or off change logging. That is done by the **changeLogging** option.

Default = **on**

changeLogMaxAge *nnn*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	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	LDBM	SDBM	GDBM	CDBM	EXOP
				X		

Specifies the maximum number of entries that the change log can contain. If the number of change log entries exceeds this value and **changeLogging off** is not specified, change log entries with the lowest change numbers are deleted. If the change log is DB2-based, change log entries are deleted until the number of remaining entries is 95% of the maximum. If the change log is file-based, change log entries are deleted until the number of remaining entries is the maximum. The value must be between 0 and 2147483647. A value of 0 indicates that there is no maximum.

Default = 0

commitCheckpointEntries *nnn*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
		X		X	X	

Specifies the maximum number of entries in the checkpoint file. An entry is added to the LDBM, CDBM, or file-based GDBM checkpoint file each time a directory entry is added, changed, deleted, or renamed. When the maximum number is reached, the entries in the checkpoint file are merged into the database file and the entries are removed from the checkpoint file. The value must be between 0 and 2147483647. A value of 0 indicates that there is no maximum.

Default = 10000

commitCheckpointTOD *hh:mm*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
		X		X	X	

Specifies a time of day at which the checkpoint file is merged into the database file. An entry is added to the LDBM, CDBM, or file-based GDBM checkpoint file each time a directory entry is added, changed, deleted, or renamed. Every day at the specified time, the entries in the checkpoint file

are merged into the database file and the entries are removed from the checkpoint file. The value must be between 00:00 and 23:59. Specify a value outside this range to disable time of day checkpoint processing.

Default = 00:00

commThreads *num-threads*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	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. You might want to have the **commThreads** set to approximately two times the number of processors that are running in your LPAR. However, this is a general rule depending upon the activity that your LDAP server experiences.

Default = 10

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

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

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X	X	X	X	X

Marks the beginning of a new database section. All global options must appear before the first database section. All options after the **database** option pertain to this backend until another **database** option is encountered.

- For *dbtype*:
  - Specify **tdbm** (DB2-based), **ldbm** (file-based), **sdbm** (RACF-based), **gdbm** (DB2-based or file-based), **cdbm** (file-based), or **exop** (extended operations).

    Notes:
    1. The server compatibility level must be at least **5** when the CDBM backend is configured. See "serverCompatLevel {3 | 4 | 5 | 6 | 7| 8}" on page 125 for more information about the **serverCompatLevel** configuration option.
    2. The EXOP backend is deprecated.
- 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 **GLDBTD31/GLDBTD64** when *dbtype* is **tdbm**, **GLDBLD31/GLDBLD64** when *dbtype* is **ldbm**, **GLDBSD31/GLDBSD64** when *dbtype* is **sdbm**, **GLDBGD31/GLDBGD64** when *dbtype* is **gdbm**, **GLDBCD31/GLDBCD64** when *dbtype* is **cdbm**, and **GLDXPD31/GLDXPD64** when *dbtype* is **exop**.

    Notes:
    1. Both DLL names must be specified for *dblibpath* as shown above. For example, to use the SDBM backend, specify the following in the LDAP server configuration file:

```
database sdbm GLDBSD31/GLDBSD64
```

2. In the job log, the LDAP server writes the DLL name that is loaded by the LDAP server. For example, if the LDAP server is run in 31-bit mode with the SDBM backend enabled, the following is written to the job log:

   ```
   database sdbm GLDBSD31 SDBM-0003
   ```

   If the LDAP server is run in 64-bit mode with the SDBM backend enabled, the following is written to the job log:

   ```
   database sdbm GLDBSD64 SDBM-0003
   ```

- For *name*:
  - This value is a name that is used to identify this backend. You cannot specify **schema, rootDSE**, or **Monitor** as the name. A name is generated if no name is specified for a backend. However, a name must be specified if the **multiserver on** option is specified for this backend and the name must not be longer than 8 characters. In addition, when multi-server mode is active, the same name must be specified for each instance of the backend within the cross-system group.

databaseDirectory *name*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
		X		X	X	

Specifies the name of the directory containing the data files that are used by the backend to store directory data. A fully qualified directory path must be specified. A unique directory must be specified for each backend. In addition, when multi-server mode is active, the same directory path must be specified for each instance of the backend within the cross-system group.

LDBM Default = /var/ldap/ldbm

GDBM Default = /var/ldap/gdbm

CDBM Default = /var/ldap/schema if **schemaPath** not specified, else **schemaPath** option setting

dbuserid *userid*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X			X		

Specifies a z/OS user ID that is the owner of the DB2 tables. When specified in a GDBM backend section, this option indicates that the GDBM backend is DB2-based and not file-based.

Note: The **dbuserid** value must be unique within the configuration file. Multiple backends on an LDAP server cannot share a database.

db2StartUpRetryInterval *num-seconds*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the number of seconds the LDAP server waits before each DB2 connection retry attempt as a consequence of the initial DB2 connection failure.

During LDAP initialization, an initial attempt at establishing a DB2 connection is made if at least one DB2based backend is defined. If the connection attempt is unsuccessful and the LDAP server is set up to wait for DB2, the LDAP server retries the connection for a specified number of times, waiting for **db2StartUpRetryInterval** seconds before each retry attempt. While waiting for a connection to DB2, the LDAP server does not receive requests. The value must be between 1 and 999.

Notes:
1. This option is ignored if no DB2-based backend (TDBM and DB2-based GDBM) is defined or if the **db2StartUpRetryLimit** configuration option has a zero value or is not specified.
2. When the server is started as a transition server (started in transition mode), this configuration option is ignored and the server ends if DB2 termination is detected before transition completes. Once transition completes, the setting in the configuration file takes effect. See "Updating LDAP configurations settings in a sysplex without server outage" on page 210.

Default = 45

db2StartUpRetryLimit *num-retries*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies a limit of the number of DB2 connection retries the LDAP server attempts as a result of the initial DB2 connection failure.

During LDAP initialization, an initial attempt at establishing a DB2 connection is made if at least one DB2-based backend is defined. If the connection attempt is unsuccessful and **db2StartUpRetryLimit** has a nonzero value, the LDAP server retries the connection for the specified **db2StartUpRetryLimit** times, waiting for the specified **db2StartUpRetryInterval** number of seconds before each retry attempt. When the number of retry attempts equals **db2StartUpRetryLimit** and a connection to DB2 still cannot be established, all backends that require DB2 fail to configure. While waiting for a connection to DB2, the LDAP server does not receive requests. The value must be between 0 and 99. A value of 0 indicates that no DB2 connection retries are to be attempted.

Notes:
1. This option is ignored if no DB2-based backend (TDBM and DB2-based GDBM) is defined.
2. When the server is started as a transition server (started in transition mode), this configuration option is ignored and set to 0 until the transition completes. Once transition completes, the setting in the configuration file takes effect. For more information about the transition server, "Updating LDAP configurations settings in a sysplex without server outage" on page 210.

Default = 0

db2Terminate {terminate | recover | restore}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies how the LDAP server reacts to a termination of DB2 after the server successfully starts.

If set to **terminate**, the LDAP server shuts down.

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

Notes:

1. This option is ignored and no DB2 monitoring is done if no DB2-based backend (TDBM and DB2-based GDBM) is configured.

   If using a sysplex distributor, this configuration option is set to **terminate**. This allows client requests to be routed to other LDAP servers in the sysplex who can connect to their databases.
2. When the server is started as a transition server (started in transition mode), this configuration option is ignored and the server ends if DB2 termination is detected before transition completes. Once transition completes, the setting in the configuration file takes effect. See "Updating LDAP configurations settings in a sysplex without server outage" on page 210.

Default = **recover**

digestRealm *hostname*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	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. Make sure that the *hostname* is a DNS-host name and not an IP address.

Default = fully qualified host name 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	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the maximum number of entries to store in the Distinguished Name normalization cache. This cache holds information that is 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	LDBM	SDBM	GDBM	CDBM	EXOP
	X			X		

Specifies the maximum number of entries to store in the Distinguished Name to Entry Identifier mapping cache. This cache holds information that is 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.

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

Default = 1000

dsnaoini *dsname*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	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 21 in "Getting DB2 installed and set up for CLI and ODBC" on page 19. This must be either a fully qualified data set name, a DD name, or a path name. A data set name is not enclosed in quotation marks or prefixed with '//', a DD name starts with '//:', and a path name starts with '/' or './'.

There are three ways to specify the CLI initialization file and the search order is as follows:

1. The DSNAOINI DD statement in the JCL for the LDAP server started task
2. The DSNAOINI environment variable
3. The **dsnaoini** configuration option. If the **dsnaoini** configuration option is specified for a backend, the option must also be specified, with the same value, for all the TDBM and DB2-based GDBM backends in the configuration file.

"Running the LDAP server using data sets" on page 176 gives more information about this process. See the DB2 information in IBM Information Management Software for z/OS Solutions Information Center 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.

enableResources {on | off}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
			X			

Specifies whether the SDBM backend supports operations on RACF resources and classes. If **on**, SDBM accepts operations for the setropts,

class, and resource profile entries. LDAP also accepts requests for creating a change log entry for a change to a RACF resource profile. If **off**, an SDBM search from the suffix does not return these entries and operations (including a change log request) involving these entries are rejected.

Default = **off**

entryCacheSize *num-entries*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X			X		

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

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

Default = 5000

entryOwnerCacheSize *num-entries*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	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	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X			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 directory bind processing in addition to the backend in which the bind DN exists. If this option is **off**, group memberships are not gathered from this backend unless the bind DN exists in this backend.

See “Associating DNs, access groups, and additional bind and directory entry access information with a bound user” on page 457 for information about group gathering after a successful bind.

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 exists. See Appendix C, “Supported server controls,” on page 679 for more information.

This option applies only to the backend in which it is defined.

Default = **off**

fileTerminate {terminate | recover}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
		X		X	X	

Specifies whether the LDAP server ends when file system errors occur. If **terminate**, the LDAP server ends when a file system error is detected. If **recover**, the LDAP server continues processing, but the backend experiencing the file system error is set to read-only mode. No updates can be made to the directory controlled by this backend. When the problem is corrected, the backend can be reset to read/write mode using the LDAP server BACKEND operator modify command. See "LDAP server operator commands" on page 205 for information about the LDAP server BACKEND modify command.

Default = **recover**

Note:
When the server is started as a transition server (started in transition mode), this configuration option is ignored and the server ends if a file system error is detected before transition completes. Once transition completes, the setting in the configuration file takes effect. See "Updating LDAP configurations settings in a sysplex without server outage" on page 210.

filterCacheBypassLimit *num-entries*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X		X	X	

Specifies the maximum number of returned entries that are allowed in the result set of any individual search that is stored in the Search Filter cache. Search filters that match more than this number of entries are not 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.

The value must be in the range of 1 to 250. This option is ignored when the filter cache is not in use.

Default = 100.

filterCacheSize *num-filters*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X		X	X	

Specifies the maximum number of filters to store in the Search Filter cache. This cache holds information that is 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 that return more entries than specified in the **filterCacheBypassLimit** option are not placed in the cache.

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

TDBM Default = 5000
LDBM Default = 5000
CDBM Default = 5000
GDBM Default = 0

idleConnectionTimeout *num-seconds*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the amount of time in seconds that the LDAP server waits for an idle connection or an idle paged search result set. When an idle connection times out, the client connection is dropped. When an idle paged search result set times out, the paged search result set is abandoned. Idle connections and idle paged search result sets are detected by the LDAP servers network monitor task, which checks for them every 30 seconds. Therefore, it is possible for an idle connection or idle paged search result set to remain active slightly longer than the **idleConnectionTimeout** value.

The value must be either 0 or between 30 and 2147483647. A value of 0 indicates that an idle connection or idle paged search results remains active indefinitely.

Default = 0 (indefinitely)

Suggested value = 1800 (30 minutes)

include *filename* **[***systemName***]**

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X	X	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.

See “Specifying a value for filename” on page 86 for information about specifying *filename*.

Note that the LDAP server does not detect loop conditions in a set of included files. Configuration might 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.

If the system name is specified, the include file is processed only on that system. This allows the LDAP server configuration files to be shared by multiple servers where each server runs on a different z/OS system. System-specific configuration information can then be placed in an include file that is processed only on the system that it applies.

krbIdentityMap {on | off}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X	X		X	

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

Default = **off**

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

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the Kerberos key table that is used by the LDAP server. The key table is used to obtain the encryption key for the Kerberos principal that is associated with the LDAP server. A key table must be provided if Kerberos authentication is used and the Kerberos KDC is not running on the same system as the LDAP server. However, a key table is not necessary if the Kerberos KDC is running on the same system as the LDAP server, the user ID associated with the LDAP server has a RACF KERB segment containing the server principal name, and the user ID associated with the LDAP server has read permission to the IRR.RUSERMAP facility class when the **KRB5_SERVER_KEYTAB** environment variable in the security server configuration file (**krb5.conf**) is set to 1. In these cases, the **krbKeytab** option is either omitted or set to none. Following is an example:

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

Default = no value

krbLDAPAdmin *kerberosIdentityDN*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the Kerberos identity that represents the LDAP root administrator. This option allows the root 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**. The **ibm-kn** attribute type is case-sensitive and must match the actual Kerberos identity. Following is an example:

```
krbLDAPAdmin ibm-kn=LDAPAdmin@MYREALM.COM
```

For information about specifying a value for a distinguished name for this option, see “Specifying a value for a distinguished name” on page 86.

listen *ldap_URL*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

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

Note the **listen** value might be established in the configuration file, or it might be established using the **-l** command-line parameter when starting the LDAP server (see “Setting up and running the LDAP server” on page 171).

Default = The server listens on all available and active IPv4 addresses, using port 389. This is equivalent to `ldap://:389`.

The format of *ldap_URL* for the **listen** option to listen on a TCP/IP socket interface is the following. This format is also used for other configuration options whose value is in LDAP URL format, such as **altServer, masterServer**, and **referral**.

{**ldap://** | **ldaps://**}[*IP_address* | *hostname* | **INADDR_ANY** | **in6addr_any**][:*portNumber*]

The format of *ldap_URL* for the **listen** option to listen on the Program Call interface is the following:

ldap://:pc

where:

ldap:// Specifies that the server listen on nonsecure addresses or ports. Note if SSL/TLS is configured for the server, then once a connection is established, the client might switch to secure communication using the **Start TLS** extended operation. Consider specifying **INADDR_ANY** or **in6addr_any** (see below), as this allows the z/OS Communications Server to determine the active interfaces rather than the LDAP server. This is preferable, especially in CINET environments with multiple TCP/IP stacks.

ldaps://
Specifies that the server listen on secure addresses or ports. When a connection is established to the server, the client must begin the SSL/TLS handshake protocol. The **sslKeyRingFile** option must also be specified when using this format. Consider specifying **INADDR_ANY** or **in6addr_any** (see below), as this allows the z/OS Communications Server to determine the active interfaces rather than the LDAP server. This is preferable, especially in CINET environments with multiple TCP/IP stacks.

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 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 active and available IP addresses.

INADDR_ANY
Specifies the **INADDR_ANY** interface. If specified, the z/OS Communications Server determines the active and available IPv4 TCP/IP interfaces on the system that the LDAP server binds and listens for requests. See *z/OS V2R2.0 Communications Server: IP Configuration Guide* and *z/OS V2R2.0 Communications Server: IP Configuration Reference* for more information about the **INADDR_ANY** interface.

in6addr_any
Specifies the **in6addr_any** interface. If specified, the z/OS Communications Server determines the active and available IPv4 and IPv6 TCP/IP interfaces on the system that the LDAP server binds and listens for requests. See *z/OS V2R2.0 Communications*

Server: IP Configuration Guide and *z/OS V2R2.0 Communications Server: IP Configuration Reference* for more information about the **in6addr_any** interface.

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 **serverSysplexGroup** option is present in the configuration file, the port number that is specified for this server instance must be the same as the port number specified for all other members of the sysplex group for dynamic workload balancing to function properly.

It is advisable to reserve the port number or numbers that are chosen here in your TCP/IP profile data set. Also, be aware that port numbers below 1024 might require more specifications. See *z/OS V2R2.0 Communications Server: IP Configuration Guide* and *z/OS V2R2.0 Communications Server: IP Configuration Reference* for more information.

pc Specifies that the LDAP server listens for program call (PC) calls from RACF change logging using the z/OS System Authorization Facility (SAF) interface. Only one LDAP server on a system can listen for PC calls.

Note 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 and you cannot specify **ldaps**.

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

- If you specify:
  ```
  ldap://
  ```
  the LDAP server binds and listens on all active and available IPv4 addresses on the system on the nonsecure default port of 389 for incoming client requests. Note this is not the same as `ldap://INADDR_ANY`, which listens specifically on the **INADDR_ANY** interface on the nonsecure default port of 389, or the `ldap://in6addr_any`, which listens specifically on the **in6addr_any** interface on the nonsecure default port of 389.
- If you specify:
  ```
  ldap://us.endicott.ibm.com:489
  ```
  the LDAP server binds and listens on all active and available IPv4 and IPv6 addresses associated with the 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 on all active and available IPv4 and IPv6 addresses associated with the host name `us.endicott.ibm.com` on the default secure port of `636` for incoming client requests.

- 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` for incoming client requests.

- If you specify:

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

the LDAP server binds and listens on all active and available IPv4 addresses on the system on the nonsecure port of `489` for incoming client requests. Note that this is not the same as `ldap://INADDR_ANY:489`, which listens specifically on the **INADDR_ANY** interface on the nonsecure port of `489`, or `ldap://in6addr_any:489`, which listens specifically on the **in6addr_any** interface on the nonsecure port of `489`.

- If you specify:

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

the LDAP server binds and listens on all active and available IPv4 addresses on the system on the secure port of `777` for incoming client requests. Note that this is not the same as `ldaps://INADDR_ANY:777`, which listens specifically on the **INADDR_ANY** interface on the secure port of `777`, or `ldaps://in6addr_any:777`, which listens specifically on the **in6addr_any** interface on the secure port of `777`.

- 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 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 `::ffff:9.130.77.75` on the secure port of `777` for incoming client requests.

- If you specify:

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

the LDAP server binds and listens on all active and available IPv4 and IPv6 addresses on the system on the nonsecure default port of `389` for incoming client requests. Note this is not the same as `ldap://INADDR_ANY`, which listens specifically on the **INADDR_ANY** interface on the nonsecure default port of `389`, or `ldap://in6addr_any`, which listens specifically on the **in6addr_any** interface on the nonsecure default port of `389`.

- If you specify:

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

the LDAP server binds and listens for PC calls from RACF change logging using the SAF interface in to the server.

Note: The **listen** parameter deprecates the **security**, **port**, and **securePort** options in the configuration file. If there is a **listen** option that is 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 is configured to listen on the port numbers specified for **securePort**, **port**, or both depending upon the **security** setting. However, configure the LDAP server using the **listen** configuration option.

logfile *filename*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the location of the file where the activity log is written when logging is enabled. See "Activity logging" on page 193 for more information.

See "Specifying a value for filename" on page 86 for information about specifying the *filename*.

Default =/etc/ldap/gldlog.output

logFileFilter *filter*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies a client IP address filter that is used to determine the activity that is included or excluded from being logged in the activity log file. Client requests originating from IP addresses allowed by the filter are written to the activity log file specified in the **logfile** configuration option.

The only supported activity log filters are ones using the **ibm-filterIP** attribute type to designate the client IPv4 addresses or IPv6 addresses with no brackets that are to be included or excluded from the activity log file. Host names and subnet masks are not supported in these filters. See "Activity logging" on page 193 for more information.

Default = ibm-filterIP=*

logFileMicroseconds {on | off}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Controls all generated log records containing microseconds in their time stamps. This setting cannot be modified by a **LOG** operator **modify** command. The default does not include microseconds in the time stamps. If **on**, all activity log time stamps include microseconds. If **off**, microseconds are not included.

Default = **off**

The **GLDLOG_MICROSECONDS** environment variable, which can also control this behavior, is now deprecated. If both the keyword and the environment variable are set, the keyword setting is used.

logFileMsgs {msgs | noMsgs}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Controls log records that are generated when messages are created by the LDAP server. The supported options are:

msgs Messages that are generated by the LDAP server are written to the activity login in addition to the normal target.

noMsgs
Indicates that messages generated by the LDAP server are not written to the activity log.

Default = **noMsgs**

The **GLDLOG_MSG** environment variable, which can also control this behavior, is now deprecated. If both the keyword and the environment variable are set, the keyword setting is used.

logFileOps {writeOps | allOps | summary | noOps}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Controls which operations generate log records. The supported options are:

writeOps
Log records are created for **add, delete, modify, modrdn**, and extended operations.

allOps
Log records are created for **writeops, bind, search, compare, abandon**, and **unbind** operations.

summary
Summary statistics are logged every hour. If any logging is collected, summary data is created hourly.

noOps
No log record is created.

Default = **noOps**

The **GLDLOG_OPS** environment variable, which can also control this behavior, is now deprecated. If both the keyword and the environment variable are set, the keyword setting is used.

logFileRecordType {begin | both | mergedRecord}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Controls when log records are generated. The supported options are:

begin Log records are created at the beginning of requests.

both Log records are created at the beginning and the end of requests.

mergedRecord
mergedRecord log records are created for all operations that are logged. These records are generated when the operation ends and contain additional information fields that are also provided in the audit log.

Default = **begin**

The **GLDLOG_TIME** environment variable, which can also control this behavior, is now deprecated. If both the keyword and the environment variable are set, the keyword setting is used.

Note: If set when using the **GLDLOG_TIME** environment variable, the possible values are **time, notime**, and **mergedRecord**.

logFileRolloverDirectory *name*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the name of the z/OS UNIX System Services file system directory where the activity log files are archived or rolled over or the Generated Data Group (GDG) base data set. If a z/OS UNIX System Services file system directory is specified, it must be a fully qualified directory path. If a GDG base data set name is specified, the **logfile** configuration option must specify the same GDG base data set name. If the **logfile** configuration option specifies a file that exists in a z/OS UNIX System Services file system directory and this option is not specified, the archived or rolled over activity log file is kept in the same directory. See "Activity logging" on page 193 for more information about activity log archiving or roll over.

Default = Directory that is specified by the **logfile** configuration option.

logFileRolloverSize *nnn*[**K** | **M** | **G**]

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the maximum size in bytes of the activity log file. When the maximum size is reached, the activity log file is rolled over or archived. The value *nnn* can be followed by a **K, M**, or **G** to indicate kilobytes, megabytes, or gigabytes, in that order, and must be at least 10K (10240) but no larger than:

18446744073709551615

18014398509481983K

17592186044415M

17179869183G

Specify 0 to disable activity log file archiving or roll over based on size. See "Activity logging" on page 193 for more information about activity log archiving or roll over.

Default = 0

logFileRolloverTOD *hh:mm*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the time of day when the activity log file is archived or rolled over. Every day at the specified time, the current activity log file is rolled over or archived. The value must be between 00:00 and 23:59. Specify a value outside this range to disable activity log file archiving or roll over based on time of day. See "Activity logging" on page 193 for more information about activity log archiving or roll over.

Default = 24:00

logFileVersion {0 | 1}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the activity log version. The versions are 0 and 1. See "Activity logging" on page 193 for more information about the activity log version.

Default = 0

masterServer *ldap_URL*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X				

Specifies for this backend the location of this replicas master server for basic replication. There is no required format for the value, however the z/OS LDAP client can only follow a **masterServer** value if it is in LDAP URL format. See "listen ldap_URL" on page 104 for a description of LDAP URL format. The presence of this option indicates that this LDAP server is a basic replication read-only replica for this backend and receives updates from a master LDAP server. Any other update requests for this backend received directly by the LDAP server is redirected to the master server. You must also specify the **masterServerDN** option in this section of the configuration file. The master server must contain all the suffixes that are defined for this backend.

The **masterServer** option can be specified multiple times if there are multiple master servers. In this case, the LDAP client attempts to contact each server in the list until it is able to establish a connection with one of the servers.

The **masterServer** option indicates that basic replication is configured for this backend section. Therefore, the **masterServer** configuration option cannot be specified if the **useAdvancedReplication** configuration option is set to **on** in the CDBM backend database section.

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	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X				

Specifies the distinguished name (DN) can always make updates to this basic replication read-only replica backend. The value must be in DN format that is described in Chapter 14, "Data model," on page 271. The presence of this option indicates that this LDAP server is a read-only replica for this backend and receives updates from a master LDAP server using the specified DN. The specified DN is a special entry that is only used when replicating to this read-only replica backend. The DN has unrestricted update, compare, and search access for all entries in the backend on this server, even if the LDAP server is in maintenance mode. When in maintenance mode, only this DN and an LDAP root administrator can access and update the entries in this backend. All other update operations for this backend received by the replica server are redirected to the master server. Care must be taken when updating this backend to ensure that the replica server remains synchronized with the master server.

You must also specify the **masterServer** option in this section of the configuration file. You cannot specify the **peerServerDN** option.

The **masterServerDN** option indicates that basic replication is configured for this backend section. Therefore, the **masterServerDN** configuration option cannot be specified if the **useAdvancedReplication** configuration option is set to **on** in the CDBM backend database section.

You might want the DN to have the same suffix as one of the **suffix** option values in the configuration file. "Establishing the root administrator DN and basic replication replica server DN and passwords" on page 151 describes how to set up your master server DN.

For information about specifying a value for a distinguished name for this option, see "Specifying a value for a distinguished name" on page 86.

masterServerPW *string*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X				

Specifies the password for the **masterServerDN** that can make updates for this backend. This option is only applicable for a basic replication read-only LDAP server. See "Establishing the root administrator DN and basic replication replica server DN and passwords" on page 151 for additional information about the master server password.

Note:

1. Use of the **masterServerPW** configuration option is discouraged in production environments. Instead, specify your **masterServerDN** as the distinguished name of an existing entry in the directory information tree, including a **userPassword** attribute. This eliminates passwords from the configuration file.
2. Password policy does not apply to the entry specified in the **masterServerDN** configuration option when the password is specified in the **masterServerPW** configuration option.

Note:

The **masterServerPW** option indicates that basic replication is configured for this backend section. Therefore, the **masterServerPW** configuration option cannot be specified if the **useAdvancedReplication** configuration option is set to **on** in the CDBM backend database section.

maxConnections *num-connections*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the maximum number of concurrently connected clients that the LDAP server allows.

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 that are 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 that is specified for this option takes into account that the server uses approximately 10 descriptors for internal functions and uses more depending upon the number of additional sockets that are 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 FILEPROCMAX option on the RACF **altuser** command are used to set the limit. Only processes with superuser authority can adjust the limit beyond the limit that is specified by MAXFILEPROC and FILEPROCMAX. Attempts to exceed this limit by non-superuser processes might be audited by the security manager.
- The maximum number of sockets that are 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.

multiserver {on | off}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X		X	X	

Indicates the operating mode that the LDAP server runs for this backend. Specifying **on** indicates the server runs in multi-server mode for this backend (see "Determining operational mode" on page 146). In multi-server mode, the LDAP server shares directory data with other instances of the LDAP server running within the sysplex. The

serverSyplexGroup configuration option must also be specified when running in multi-server mode. Specifying **off** indicates the server runs in single-server mode for this backend.

Default = **off**

You can configure a backend to operate in single-server mode while another backend operates in multi-server mode except when GDBM or CDBM is configured. When CDBM or GDBM is configured, all TDBM, LDBM, GDBM, and CDBM backends must be configured to use the same operating mode.

nativeAuthSubtree {all | *dn*}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X			X	

Specifies the distinguished name of a subtree where all of its entries are eligible to participate in native authentication. This option can appear multiple times to specify all subtrees that use native authentication. If this option is omitted 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.

For information about specifying a value for a distinguished name for this option, see "Specifying a value for a distinguished name" on page 86.

Default = **all**

nativeUpdateAllowed {on | off | reset}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X			X	

When set to **on** or **reset**, enables native password or password phrase changes in the security server to occur through a modify request to the TDBM, LDBM, or CDBM backend if the **useNativeAuth selected** or **all** option is specified.

When set to **reset**, this option also allows a bind to the backend to succeed even if the specified native authentication password is expired, if the **PasswordPolicy** control is included in the bind request. After the bind, only the special delete-add modification of the bound user's **userPassword** attribute can be performed to reset the native authentication password. Once complete, other LDAP operations can be performed.

This option does not affect the ability to change a native password or password phrase during a bind operation.

Note: z/OS LDAP password policy does not apply to entries participating in native authentication.

Default = **off**

operationsMonitor {ip | ipAny | all}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the search patterns that are monitored by the LDAP server. Operations monitor supports search patterns of **searchStats** and **searchIPStats**. A **searchStats** pattern consists of the search parameters (search base, scope, filter, and attributes to be returned) and status (SUCCESS or FAILURE). A **searchIPStats** pattern consists of the same elements as in the **searchStats** pattern, but also includes the client IP address. If operations monitor is enabled, LDAP monitors search statistics for the types of search patterns that are configured. See "Operations monitor" on page 601 for more information about operations monitor.

If set to **ip**, then only **searchIPStats** patterns are monitored. This option setting is useful in determining if there are any specific clients spamming the LDAP server.

If set to **ipAny**, then only **searchStats** patterns are monitored. This option is useful for evaluating the performance of search patterns.

If set to **all**, the operations monitor monitors both **searchStats** and **searchIPStats** patterns. Therefore, each search is included in search patterns and match the **searchStats** pattern and one matches the **searchIPStats** pattern.

Default = **ipAny**

operationsMonitorSize *num-entries*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the maximum number of search patterns for which the operations monitor gathers statistics. The value must be between 0 and 2147483647. A value of 0 indicates that the operations monitor is turned off. When the operations monitor is turned off, the **cn=operations,cn=monitor** entry is not returned on a **cn=monitor** search.

Default = 1000

pcIdleConnectionTimeout *num-seconds*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the amount of time in seconds that an idle connection remains valid over the LDAP PC (program call) callable interface. After the specified time, the PC connection is considered no longer in use and any resources that are associated with the connection are released. Idle connections are detected when the LDAP server receives a new PC connection or a request on an existing PC connection.

The value must be either 0 or between 30 and 2147483647. A value of 0 indicates that an idle connection remains indefinitely.

Default = 0 (indefinitely)

Suggested value = 0

pcThreads *num-threads*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	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. No threads are used if the program call interface is not active. The value must be in the range of 2 to 2147483647.

Default = 10

peerServerDN *dn*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X				

Specifies the distinguished name (DN) that can make updates to this basic replication peer replica backend. The value must be in DN format that is described in Chapter 14, "Data model," on page 271. The presence of this option indicates that this LDAP server is a peer replica for this backend, and can receive updates from another peer LDAP server using the specified DN and processing updates that are received from clients. The specified DN is a special entry that is only used when replicating to this peer replica backend. The DN has unrestricted update, compare, and search access for all entries in the backend on this server, even if the LDAP server is in maintenance mode. When in maintenance mode, only this DN and an LDAP root administrator can access and update the entries in this backend.

Update operations for this backend received from you bound as **peerServerDN** (or as an LDAP root administrator when in maintenance mode) are performed on the local database and are not sent to any peer and read-only replica servers. When not in maintenance mode, all other update operations for this backend are performed on the local database and are sent to the other peer and read-only replica servers. Update operations from a peer or a master are never replicated. It does not matter if you are in maintenance mode or not. Updates that are made by an LDAP root administrator are replicated unless the server is in maintenance mode.

You cannot also specify the **masterServerDN** option in this section of the configuration file.

The **peerServerDN** option indicates that basic peer-to-peer replication is configured for this backend section. Therefore, the **peerServerDN** configuration option cannot be specified if the **useAdvancedReplication** configuration option is set to **on** in the CDBM backend database section.

You might want the DN to have the same suffix as one of the **suffix** option values in the configuration file. "Establishing the root administrator DN and basic replication replica server DN and passwords" on page 151 describes how to set up your peer replica DN.

For information about specifying a value for a distinguished name for this option, see "Specifying a value for a distinguished name" on page 86.

peerServerPW *string*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X				

Specifies the password for the **peerServerDN** that can make updates for this backend. This option is only applicable for a basic replication peer replica LDAP server. See "Establishing the root administrator DN and basic replication replica server DN and passwords" on page 151 for additional information about the peer server password.

Note:

1. Use of the **peerServerPW** configuration option is discouraged in production environments. Instead, specify your **peerServerDN** as the distinguished name of an existing entry in the directory information tree, including a **userPassword** attribute. This eliminates passwords from the configuration file.

   The **peerServerPW** option indicates that basic peer-to-peer replication is configured for this backend section. Therefore, the **peerServerPW** configuration option cannot be specified if the **useAdvancedReplication** configuration option is set to **on** in the CDBM backend database section.
2. Password policy does not apply to the entry specified in the **peerServerDN** configuration option when the password is specified in the **peerServerPW** configuration option.

persistentSearch {on | off}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X		X	X	

Allows or disallows persistent search for changes that are made to entries in a backend. When **off** is specified, persistent search requests for this backend are rejected. See "PersistentSearch" on page 685 for more information about persistent search.

Default = **off**

plugin *pluginType pluginName pluginInit* **[***pluginParameters***]**

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Defines a plug-in extension to the LDAP server. Writing an LDAP server plug-in and using the SLAPI service routines are described in *z/OS IBM Tivoli Directory Server Plug-in Reference for z/OS*. A sample plug-in and its makefile are included in **/usr/lpp/ldap/examples**. Building and using the sample plug-in are described in the *z/OS IBM Tivoli Directory Server Plug-in Reference for z/OS*.

- For *pluginType*:
  - Specify **preOperation, clientOperation**, or **postOperation**. A **preOperation** plug-in is called by the LDAP server before a client request is processed. A **clientOperation** plug-in is called to process a client request. A **postOperation** plug-in is called after a client request is processed. A **clientOperation** plug-in is called when a client request matches a distinguished name suffix or extended operation object identifier that is registered for the plug-in.
- For *pluginName*:
  - Specify the name of the shared library (DLL) containing the plug-in code. A plug-in that supports both 31-bit and 64-bit addressing modes

specifies both file names that are separated by a slash, "/", such as `plugin31/plugin64`. A plug-in that supports only 31-bit addressing mode specifies one file name, such as `plugin31`.

- For *pluginInit*:
  - Specify the name of the plug-in initialization routine. This plug-in routine is called by the LDAP server to allow the plug-in to initialize. The plug-in initialization routine registers supported message types, distinguished name suffixes, and extended operation object identifiers that are supported by the plug-in.
- For *pluginParameters*:
  - Optionally, specify plug-in parameters. The plug-in can retrieve these parameters using the **slapi_pblock_get()** routine.

The ICTX and remote crypto plug-ins are plug-in extensions that are shipped by the z/OS LDAP server that provide more function.

- The ICTX plug-in allows resource managers that do not exist on z/OS to centralize authorization decisions and security event logging requests by using RACF. This enables consolidation of security authorization and auditing functions. See ICTX plug-in for more information.
- The remote crypto plug-in allows remote applications the ability to access PKCS#11 or CCA callable services that are implemented within ICSF. PKCS#11 is one of the cryptographic standards of Public-Key Cryptographic Standards (PKCS) that defines a platform-independent programming interface to cryptographic tokens. CCA is regarding the IBM Common Cryptographic Architecture. See Remote crypto plug-in for more information.

port *num-port*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Note: The **port** option has been deprecated by the **listen** option. See "listen ldap_URL" on page 104 for information about the **listen** option.

Specifies the TCP/IP port used by the LDAP server for non-SSL communications. The value must be in the range of 1 to 65535.

Default = 389

If the **serverSysplexGroup** option is present in the configuration file, the port number that is specified for this server instance must be the same as the port number specified for all other members of the sysplex group for dynamic workload balancing to function properly.

The port number might be established in the configuration file, or it might be established using the **-p** command-line parameter when starting the LDAP server (see "Setting up and running the LDAP server" on page 171).

It is advisable to reserve the port number that is chosen here in your TCP/IP profile data set. Also, be aware that port numbers below 1024 might require additional specifications. See *z/OS V2R2.0 Communications Server: IP Configuration Guide* and *z/OS V2R2.0 Communications Server: IP Configuration Reference* for further information.

pwCryptCompat {on | off}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X			X	

Specifies whether to use an EBCDIC version or a UTF-8 version of the crypt() algorithm to hash passwords when **pwEncryption crypt** is contained in this section of the configuration file. If **on**, the EBCDIC version of the crypt() algorithm is used. This is what the z/OS Integrated Security Services LDAP server used. If **off**, the UTF-8 version is used. Note ASCII is a subset of UTF-8. When sharing LDAP directory data between z/OS and an ASCII-based platform, specify **pwCryptCompat off** to ensure that the hashed value is the same on both platforms.

Default = **on**

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

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X			X	

Specifies what encryption or hashing method to use when storing the **userPassword** and **ibm-slapdAdminPw** attribute values in the backend of the directory.

none Specifies no encryption. The **userPassword** and **ibm-slapdAdminPw** attribute values are stored in clear text format. The stored values are prefixed with the tag {none}. The original value, without the tag, is returned for a search request.

crypt Specifies that **userPassword** and **ibm-slapdAdminPw** attribute values are hashed by the crypt() algorithm before they are stored in the directory. The stored values are prefixed with the tag {crypt}. The versions of the crypt() algorithm are: an EBCDIC-based version and a UTF-8-based version. See the **pwCryptCompat** option and the notes below for information about selecting which version to use. The original password value cannot be retrieved in clear text format. The tag and the hashed value are returned for a search request.

MD5 Specifies that **userPassword** and **ibm-slapdAdminPw** attribute values are hashed by the MD5 hashing algorithm before they are stored in the directory. The stored values are prefixed with the tag {MD5}. The original password value cannot be retrieved in clear text format. The tag and the hashed value are returned for a search request.

SHA Specifies that **userPassword** and **ibm-slapdAdminPw** attribute values are hashed by the SHA hashing algorithm before they are stored in the directory. The stored values are prefixed with the tag {SHA}. The original password value cannot be retrieved in clear text format. The tag and the hashed value are returned for a search request.

SSHA Specifies that **userPassword** and **ibm-slapdAdminPw** attribute values are hashed by the Salted SHA (SSHA) hashing algorithm before they are stored in the directory. The stored values are prefixed with the tag {SSHA}. The original password value cannot

be retrieved in clear text format. The tag and the base64-encoded hashed and salt values are returned for a search request.

SHA224, SHA256, SHA384, SHA512
Specifies that **userPassword** and **ibm-slapdAdminPw** attribute values are hashed by the specified SHA-2 hashing algorithm before they are stored in the directory. The stored values are prefixed with the specified tag (for example, {SHA224}). The original password value cannot be retrieved in clear text format. The tag and the base64-encoded hashed value are returned for a search request.

SSHA224, SSHA256, SSHA384, SSHA512
Specifies that **userPassword** and **ibm-slapdAdminPw** attribute values are hashed by the specified Salted SHA-2 hashing algorithm before they are stored in the directory. The stored values are prefixed with the specified tag (for example, {SSHA224}). The original password value cannot be retrieved in clear text format. The tag and the base64-encoded hashed and salt values are returned for a search request.

DES:*keylabel*
Specifies that **userPassword** and **ibm-slapdAdminPw** attribute values are encrypted by the DES algorithm before they are stored in the directory. The stored values are prefixed with the tag '{DES:*keylabel*}'. The original password value, without the tag, is returned for a search request. The key label must refer to either a valid data-encrypting key that is generated by the KGUP utility and stored in the ICSF CKDS or to an entry in the data set referenced by the LDAPKEYS DD statement. See "Symmetric encryption keys" on page 79 for more information.

AES:*keylabel*
Specifies that **userPassword** and **ibm-slapdAdminPw** attribute values are encrypted by the AES algorithm using the specified key label before they are stored in the directory. The stored values are prefixed with the tag {AES:*keylabel*}. The original password value without the tag is returned for a search request. The key label must refer to either a valid data-encrypting key that is generated by the KGUP utility and stored in the ICSF CKDS or to an entry in the data set referenced by the LDAPKEYS DD statement. See "Symmetric encryption keys" on page 79 for more information.

Note:

1. When a password is stored in a TDBM, LDBM, or CDBM 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 or hashed in that same method for password verification.
2. The crypt algorithm, which is implemented across many platforms, accepts only the first 8 characters of a password. As a result, any password that is supplied on a bind or compare operation that matches the first 8 characters of a **userPassword** attribute value that is hashed with the crypt algorithm in the directory matches.
3. When the **pwCryptCompat** option is set to **on**, the values hashed using the crypt algorithm are not portable to other X/Open-conformant systems if the **userPassword** and **ibm-slapdAdminPw** attribute values are unloaded using the **ds2ldif** utility with the **-t** command-line parameter and loaded by another platform's load utility. If the

pwCryptCompat option is set to **off**, the values hashed using the crypt algorithm are portable to other X/Open-conformant systems if the **userPassword** and **ibm-slapdAdminPw** attribute values are unloaded using the **ds2ldif** utility with the **-t** command-line parameter. The output LDIF file from **ds2ldif** can then be loaded using another platform's load utility.

4. If a tagged encrypted or hashed **userPassword** and **ibm-slapdAdminPw** attribute values is included in an add or modify operation, the attribute value is added as it is with no additional encryption or hashing performed on the value even if the **pwEncryption** configuration option is set to a different type of encryption or hashing.

Default = **none**

pwSearchOutput {binary | base64}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the format of MD5 and SHA hashed **userPassword** and **ibm-slapdAdminPw** attribute values when retrieved on a search operation. This option does not affect the retrieval of Salted SHA (SSHA), SHA-2, or Salted SHA-2 hashed **userPassword** and **ibm-slapdAdminPw** attribute values on a search operation.

If set to **binary** and **userPassword** or **ibm-slapdAdminPw** attribute value is hashed in MD5 or SHA, the LDAP server returns the encryption tag (either {MD5} or {SHA}) in UTF-8 followed by the binary hash.

If set to **base64** and **userPassword** or **ibm-slapdAdminPw** attribute value is hashed in MD5 or SHA, the LDAP server returns the encryption tag (either {MD5} or {SHA}) in UTF-8 followed by the base64-encoded binary hash.

For an example of using this option, see "One-way hashing formats" on page 77.

Default = **binary**

readOnly {on | off}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X	X	X	X	

Specifies the ability to modify the database. The LDAP server **BACKEND** operator modify command can be used to change the backend database to read/write or read-only mode while the LDAP server is running. Any attempt to use the LDAP server to modify the database fails if **readOnly** is turned **on**.

Note:

1. For GDBM, change log entries are not created and are not trimmed (deleted) by the LDAP server when **readOnly** is **on**.
2. When running in multi-server mode, the **readOnly** configuration option is the same for all LDAP servers in the cross-system group because any LDAP server can potentially handle update requests.

3. For SDBM, **readonly on** does not prevent changing a RACF password during a bind operation, using the *currentvalue/newvalue* format. However, it does prevent changing the password by using a modify operation of the **racfpassword** attribute.
4. When LDBM, TDBM, or CDBM is using native authentication, the RACF password can be changed during bind even though **readonly on** is specified. The RACF password cannot be changed by using the LDBM, TDBM, or CDBM native authentication modify of the **userpassword** attribute.
5. If authenticating or comparing an LDBM, TDBM, or CDBM entry that is subject to password policy in the LDAP server, **readonly on** does not prevent the password policy operational attributes from being updated in the entry.

Default = **off**

referral *ldap_URL*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the referral to pass back when the target of a client request is not included in any suffix within the LDAP server. It is also known as the default referral. The **referral** option can appear multiple times and lists equivalent servers. There is no required format for the value, however, the z/OS LDAP client can only follow a referral value if it is in LDAP URL format. See "listen ldap_URL" on page 104 for a description of LDAP URL format.

A default referral is not returned to the client if the client request includes the **manageDsaIT** control. See "manageDsaIT" on page 682 for more information about this control.

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

schemaPath *name*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the name of the file directory containing the LDAP schema database. A fully qualified directory path must be specified. When multi-server mode is active, the same schema path must be specified for each LDAP server within the cross-system group. The schema database file is automatically created during LDAP server initialization if it does not exist. The LDAP server must have write access to the schema directory. This configuration option also determines the directory that is used by CDBM to store its data if the CDBM backend is configured and the **databaseDirectory** configuration option is not specified in the CDBM backend configuration section.

Default = /var/ldap/schema

schemaReplaceByValue {on | off}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
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 typical modify operation: all the values currently in the attribute are replaced by the values that are 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. Except in several specific cases, the values of the attribute that are in the initial LDAP server schema cannot be changed or removed. See "Updating the schema" on page 297 for more information about 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**

secretEncryption {none | DES:*keylabel* **| AES:***keylabel***}**

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X			X	

Specifies the encryption method to use when storing the **secretKey, replicaCredentials, ibm-replicaKeyPwd**, and **ibm-slapdMasterPw** attribute values in this backend. Applications might use the **secretKey** attribute type to store sensitive data that must be encrypted in the directory and to retrieve the data in clear text format. This encryption method is used to protect the **replicaCredentials** attribute values in this backend when basic replication is enabled. This encryption method also protects the **ibm-replicaKeyPwd** and **ibm-slapdMasterPw** attribute values in this backend when advanced replication is enabled.

none Specifies no encryption. The **secretKey, replicaCredentials, ibm-replicaKeyPwd**, and **ibm-slapdMasterPw** attribute value is stored in clear text format. The stored value is prefixed with the tag {none}. This is the default if the **secretEncryption** option is not specified. The attribute value without the tag is returned for a search request.

DES:*keylabel*
The **secretKey, replicaCredentials, ibm-replicaKeyPwd**, and **ibm-slapdMasterPw** attribute value is encrypted by the DES algorithm before it is stored in the directory. The stored value is prefixed with the tag {DES:keylabel}. The original value without the tag is returned for a search request. The key label must refer to either a valid data-encrypting key that is generated by the KGUP utility and stored in the ICSF CKDS or to an entry in the data set referenced by the LDAPKEYS DD statement. See "Symmetric encryption keys" on page 79 for more information.

AES:*keylabel*

The **secretKey, replicaCredentials, ibm-replicaKeyPwd**, and **ibm-slapdMasterPw** attribute value is encrypted by the AES algorithm before it is stored in the directory. The stored value is prefixed with the tag {AES:keylabel}. The original value without the tag is returned for a search request. The key label must refer to either a valid data-encrypting key that is generated by the KGUP utility and stored in the ICSF CKDS or to an entry in the data set referenced by the LDAPKEYS DD statement. See "Symmetric encryption keys" on page 79 for more information.

Default = **none**

securePort *num-port*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Note: The **securePort** option has been deprecated by the **listen** option. See "listen ldap_URL" on page 104 for information about the **listen** option.

Specifies the TCP/IP port that is used by the LDAP server for SSL communications. The value must be in the range of 1 to 65535.

Default = 636

If the **serverSysplexGroup** option is present in the configuration file, the secure port number that is specified for this server instance must be the same as the secure port number specified for all other members of the sysplex group for dynamic workload balancing to function properly.

The secure port number might be established in the configuration file, or it might be established using the **-s** command-line parameter when starting the LDAP server (see "Setting up and running the LDAP server" on page 171).

It is advisable to reserve the port number that is chosen here in your TCP/IP profile data set. Also, be aware that port numbers below 1024 might require additional specifications. See *z/OS V2R2.0 Communications Server: IP Configuration Guide* and *z/OS V2R2.0 Communications Server: IP Configuration Reference* for further information.

security {ssl | sslonly | none | nossl}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Note: The **security** option has been deprecated by the **listen** option. See "listen ldap_URL" on page 104 for information about the **listen** option.

Specifies what type of communications is accepted by the LDAP server. The **ssl** setting indicates that the server listens on the secure port and the non-secure port. The **sslonly** setting means that the server listens only on the secure port. The **none** or **nossl** settings indicate that the server listens only on the non-secure port. The **sslKeyRingFile** option must also be specified when the **ssl** or **sslonly** settings are used.

Default = **none**

securityLabel {on | off}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Determines if the security label processing is activated with bound LDAP clients. When **on**, the security labels that are associated with the LDAP client and LDAP server are verified during the authentication process. Security labels are recorded in all LDAP audit records. When **off**, no security label processing is done.

Default = **off**

Use this option when configuring the LDAP server in a multilevel security environment. For more information about configuring a z/OS system for multilevel security and how to configure an LDAP server in that environment, see *z/OS Planning for Multilevel Security and the Common Criteria*.

sendV3stringsoverV2as {UTF-8 | ISO8859-1}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	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 597 for more detailed information about the use of this setting.

serverCompatLevel {3 | 4 | 5 | 6 | 7| 8}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the server compatibility level. This value can be used to limit the functions that are supported by the server so that the server can be compatible with older versions of LDAP servers when they are sharing directory data in a sysplex group. To produce consistent results, all the LDAP servers in the same sysplex group name must support the same functions. If fallback is required to a lower server compatibility level than is being used, it is necessary to remove all exploitation of function that is available at the current compatibility level but not at the lower level. The server might not start at the lower level until this is complete. If fallback is necessary with a server that is using the TDBM or DB2-based GDBM backend, see "Fallback from a TDBM or DB2-based GDBM backend in z/OS IBM TDS to an earlier z/OS IBM TDS version" on page 208 for fallback procedures.

Notes:

1. If there are DB2-based backends that are configured in the LDAP server, the **serverCompatLevel** value also sets the DB_VERSION value in the DB2 DIR_MISC table for the backend. The DB_VERSION value is queried at LDAP server initialization to verify that the DB2-based backend is running on a supported level for the z/OS release.

Therefore, it is especially important to set this value appropriately when running in multi-server mode and sharing DB2-based backends to the earliest z/OS LDAP server release that is to be shared.

2. Updating the **serverCompatLevel** configuration option without a server outage is supported for z/OS Version 2 Release 2 and above. For more information about the transition server, see "Updating LDAP configurations settings in a sysplex without server outage" on page 210.
3. When the server is started as a transition server (started in transition mode), the **serverCompatLevel** of the sysplex group owner is used. Once transition completes, the setting in the configuration file takes effect.

The **serverCompatLevel** values are:

- **3** - This value limits the sharing of data in a sysplex to TDBM backends, DB2-based GDBM backends, and schema. Basic replication is supported from (but not into) the sysplex. Dynamic and nested groups are supported, as is schema replace by value. Specify this value when a z/OS Integrated Security Services (ISS) LDAP server is running in the sysplex.

  Level 3 is deprecated from z/OS Version 2 Release 2.
- **4** - This value enables cross-system coupling facility (XCF) messaging support for TDBM and DB2-based GDBM backends in the sysplex group and supports basic replication from and into the sysplex.

  Note: When the schema, LDBM, and file-based GDBM backends are shared in a sysplex, XCF messaging is used to communicate between the LDAP servers in the same sysplex group no matter the **serverCompatLevel** setting.

  Specify this value when the sysplex group contains a z/OS IBM TDS server running on z/OS V1R10 or earlier and there are no ISS LDAP servers in the sysplex.
- **5** - This value enables advanced replication and allows the CDBM backend to be configured. Schema and all backends can be shared in the sysplex. Specify this value when the sysplex group only contains z/OS IBM TDS servers running on z/OS V1R11 or later.
- **6** - This value enables ACL filters, password policy, Salted SHA (SSHA) password hashing, and usage of additional schema syntaxes and matching rules. Specify this value when the sysplex group only contains z/OS IBM TDS servers running on z/OS V1R12 or later.
- **7** - This value enables usage of group search limits, administrative roles, and hashing **userPassword** attribute values using the SHA-2 and Salted SHA-2 algorithms. It also supports hashing and encrypting **ibm-slapdAdminPw** attribute values using the same algorithms as for **userPassword** attribute values. Specify this value when the sysplex group only contains z/OS IBM TDS servers running on z/OS V1R13 or later.
- **8** - This value enables usage of the read-only replica password policy replication support. Specify this value when the sysplex group only contains z/OS IBM TDS servers running on z/OS Version 2 Release 2 or later. See "Replicating password policy operational attributes" on page 383 for more information.

Default = **8** if not running in a sysplex (the **serverSysplexGroup** configuration option is not specified).

Default = **4** if running in a sysplex (the **serverSysplexGroup** configuration option is specified)

serverEtherAddr *mac_address*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the Media Access Control (MAC) address that is 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 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 runs on this (hardware) system.

The MAC address consists of 12 hexadecimal digits. The suggested form of the *mac_address* is:

```
4xmmmmssssss
```

Where:

x Is a one-character LDAP directory number. If more than one LDAP server is operating on a processor, specify a different *x* value for each server. If more than 16 LDAP servers are wanted, then use a serial number and model number from a processor that is not running an LDAP server. If another processor 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 processor.

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

It is not necessary to follow this convention if you specify the **serverEtherAddr** option for all LDAP servers in your enterprise. In this case, you can specify any combination of 12 hexadecimal digits if each LDAP server has a unique value.

Following is an example:

```
serverEtherAddr 4A123401234D
```

Default = The LDAP server uses the hardware model and serial numbers to generate a MAC address.

serverKrbPrinc *kerberosIdentity*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the Kerberos principal name that is assigned to the LDAP server that was created in "Defining the Kerberos identity" on page 49. This value becomes the server name in Kerberos service tickets. The principal name must consist of characters that can be represented in the ISO8859-1 code page. The format for *kerberosIdentity* is:

```
ldap_prefix/primary-dns-name@krbRealmName
```

Where

ldap_prefix
Is `ldap` or `LDAP`. Use `ldap` to assure interoperability with all LDAP clients. `LDAP` is accepted, but this value is not usable with many non-z/OS LDAP clients.

primary-dns-name
Is the canonical host name returned by the DNS name service.

krbRealmName
Is the Kerberos defined realm that the LDAP server operates. For more information about setting up a Kerberos realm on z/OS, see *z/OS Integrated Security Services Network Authentication Service Administration*.

Following are examples:

```
serverKrbPrinc LDAP/myhost.realm.com@MYREALM.COM
serverKrbPrinc ldap/myhost.myrealm.com@MYREALM.COM
```

Default = **ldap/***primary-dns-hostname@default-krbRealmName*

serverName *string*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	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 that is specified by the **dsnaoini** option in the configuration file. See the DB2 information in IBM Information Management Software for z/OS Solutions Information Center for a description of the DSNAOINI ODBC initialization data set contents. Using the example DSNAOINI file in Figure 2 on page 21 the value of *string* for **serverName** is **LOC1**.

If the **serverName** configuration option is specified for a backend, the option must also be specified, with the same value, for all the TDBM and DB2-based GDBM backends in the configuration file.

Default = The default data source is used. This is the DB2 subsystem that is specified by the MVSDEFAULTSSID record in the DSNAOINI file.

serverSysplexGroup *name*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies that this LDAP server is participating in data sharing in a sysplex and indicates the name of the cross-system coupling facility (XCF) group. All LDAP servers in the sysplex that specify the same group name share the LDAP server schema and the directories of backends that specify the **multiserver on** option. The group name is 1-8 characters and consists of letters (A-Z), numbers (0-9), and special characters (@, #, and $). The special characters must be in the IBM-1047 code page.

sizeLimit *num-limit*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X	X	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 or in group search limits. Specifying this before a **database** line in the configuration file sets the option for all backends, except EXOP. Specifying it after a **database** line sets the option just for the backend that is defined by the **database** line. Specifying a size limit using group search limits sets the limit only for the members of that group. See "Managing group search limits" on page 423 for more information about group search limits.

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 determining the size limit for a search request.

- If the client has not bound as an administrator:
  - If a group search size limit exists for the requester, then the size that is used to limit the search is the smaller of the size limit that is passed by the client and the group search size limit. If the client does not specify a size limit on the search, then the group search size limit is used.
  - If a group search size limit does not exist for the requester, then the size that is used to limit the search is the smaller of the size limit that is passed by the client and the size limit that is determined by the server from the **sizeLimit** configuration options in the configuration file. If the client does not specify a size limit, then the server size limit is used.
- If the client has bound as an administrator, the size limit is the value that is passed by the client. If the client does not specify a limit, then the number of entries that are returned is unlimited. The size limits from the configuration file and from group search limits are ignored when the client has bound as an administrator.

When accessing the z/OS LDAP server support for RACF (the SDBM backend), the number of entries that are returned might be further restricted by limits that are imposed by RACF. See Chapter 17, "Accessing RACF information," on page 327 for more information:

- The limit is the smaller of the limit that is passed by the client and the limit that is read by the server from the **sizeLimit** option in the configuration file (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 that are returned might be further restricted by limits that are imposed by RACF. See Chapter 17, "Accessing RACF information," on page 327 for more information.

There are additional considerations for size limit when performing a subtree search from the root DSE (a NULL-based search). See "Root DSE search with subtree scope (Null-based subtree search)" on page 596 for more information.

srvStartUpError {terminate | ignore}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies whether the LDAP server stops if a backend or plug-in fails to initialize after the configuration file is read. If **terminate**, the server ends when any backend or plug-in fails to initialize. If **ignore**, the LDAP server continues processing if the schema successfully initializes. The option also applies to failures when initializing the LDAP PC callable support interface if that has been configured and initializing WLM support. Note a configuration error that occurs before backend or plug-in initialization begins always causes the server to end.

Note:
When the server is started as a transition server (started in transition mode), this configuration option is ignored and the server behaves as if **srvStartupError** is set to **terminate**.

Default = **terminate**

sslAuth {serverAuth | serverClientAuth}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	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 **serverClientAuth** method allows the LDAP client to validate the LDAP server. In addition, the LDAP server validates the LDAP client if the client sends its digital certificate on the initial contact between the client and the server.

Note: In order for clients to perform **SASL EXTERNAL** binds to the LDAP server, it is necessary to configure the server with **sslAuth serverClientAuth**.

See "Setting up for SSL/TLS" on page 68 for more SSL/TLS information.

Default = **serverAuth**

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

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the label of the certificate that is used for LDAP server authentication. If using a key database file, the certificate is created and managed using the **gskkyman** utility. If using a RACF key ring, the certificate is created and managed using the **RACDCERT** command. If using a PKCS #11 token, the certificate can be created and managed by

using either the **gskkyman** utility or the **RACDCERT** command. See *z/OS Cryptographic Services System SSL Programming* for details on using the **gskkyman** utility or *z/OS Security Server RACF Command Language Reference* for details on using the **RACDCERT** command. See "Setting up for SSL/TLS" on page 68 for more SSL/TLS information.

Default = **none**

If the value is **none** (by default or by specification), the default certificate, marked in the key database file, the RACF key ring, or the PKCS #11 token, is used for server authentication.

sslCipherSpecs {*string* | **GSK_V3_CIPHER_SPECS_EXPANDED** | **ANY**}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the SSL Version 3.0 and TLS Version 1.0 cipher specifications that the LDAP server accepts from clients. Use of this option to specify the specific cipher suites is limited, and provided only for compatibility with earlier versions. It supports only a portion of the cipher suites available in z/OS System SSL, contains no 4-character cipher suites, and provides no order of preference. The preferred approach is to set the option to GSK_V3_CIPHER_SPECS_EXPANDED and then set the environment variable GSK_V3_CIPHER_SPECS_EXPANDED to the list of 4-character cipher specifications you want, in order of preference.

If the cipher specifications you want are included in Table 8 on page 74 and if the order of preference matches the default order that is provided by z/OS System SSL, then the **sslCipherSpecs** option may be used with any of the values that are described.

In this case, the cipher specification is a blank delimited string that represents an ORed bit-mask indicating the SSL/TLS cipher specifications that are accepted from clients. Clients that support any of the specified cipher specifications are able to establish an SSL/TLS connection with the server. Table 8 on page 74 lists the CipherSpec mask values and the related decimal, hexadecimal, and keyword values. See *z/OS Cryptographic Services System SSL Programming* for a description of supported cipher specifications.

The cipher specification might 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_EXPORT`
  - `52992` is the same as specifying `ALL-RC2_MD5_EXPORT-RC4_MD5_EXPORT`

Depending upon the level of System SSL support installed, some ciphers might not be supported. System SSL ignores the unsupported ciphers. See the System SSL documentation to determine the specific ciphers that your installation supports.

See "Setting up for SSL/TLS" on page 68 for more SSL/TLS information.

Default = **ANY-RC4_MD5_EXPORT-RC4_MD5_US-RC4_SHA_US-RC2_MD5_EXPORT**

sslKeyRingFile *name*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the path and file name of the SSL/TLS key database file, the name of the RACF key ring, or the name of the PKCS #11 token to be used by the LDAP server. SSL/TLS connections are not available if this option is not specified.

When using a key database file, the file path and name that is specified here must match the path and name of the key database file that was created using the **gskkyman** utility (see *z/OS Cryptographic Services System SSL Programming*). Also, see "Setting up for SSL/TLS" on page 68 for more SSL/TLS 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** configuration options to use this support.

The LDAP server also supports the use of a PKCS #11 token. Specify the PKCS #11 token on the **sslKeyRingFile** configuration option in the following format (where *NAME* is the name of the PKCS #11 token): `*TOKEN*/NAME`. Ensure that the **sslKeyRingFilePW** and **sslKeyRingPWStashFile** configuration options are commented out to use this support.

See "Creating and using key databases, key rings, or PKCS #11 tokens" on page 69 for more information.

sslKeyRingFilePW *string*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	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 Cryptographic Services System SSL Programming*). Also, see "Setting up for SSL/TLS" on page 68 for more SSL/TLS information.

Note: Use of the **sslKeyRingFilePW** configuration option is discouraged. As an alternative, use either a RACF key ring, a PKCS #11 token, or specify the **sslKeyRingPWStashFile** configuration option.

Comment out the **sslKeyRingFilePW** and **sslKeyRingPWStashFile** configuration options if you are using a RACF key ring or PKCS #11 token.

sslKeyRingPWStashFile *name*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies a file system file name where the password for the servers key database file is stashed. Use the full path name of the stash file in the file system for *name*.

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 68 for more SSL/TLS information.

Comment out the **sslKeyRingFilePW** and **sslKeyRingPWStashFile** configuration options if you are using a RACF key ring or PKCS #11 token.

sslMapCertificate {off | check | add | replace} {fail | ignore}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the server maps a certificate used in a **SASL EXTERNAL** bind to the RACF user that is associated with the certificate.

When **check**, **add**, or **replace** is specified for the first value, RACF is searched for the user ID associated with the certificate used during a SASL certificate bind. The **sslKeyRingFile** configuration option must be specified to indicate which key database, RACF key ring, or PKCS #11 token to use to do this. If there is no RACF user ID associated with the certificate and **fail** is specified for the second value, then the **SASL EXTERNAL** bind fails. If there is no associated RACF user ID and **ignore** is specified for the second value, the bind continues without mapping the certificate to a RACF user.

If an associated RACF user ID is found and **add** or **replace** is specified for the first value, a distinguished name (DN) is created based on the user ID and the SDBM suffix. For **add**, this mapped DN is added to the list of DNs associated with the bind DN that was created from the subject's name in the certificate. For **replace**, this mapped DN replaces the bind DN that was created from the subject's name in the certificate. The mapped DN is used when gathering the groups in which the bound user exists and when checking authorization for LDAP operations, including SDBM operations. SDBM must be configured when **add** or **replace** is specified.

When **off** is specified for the first value, RACF is not searched for the user ID associated with the certificate and no certificate mapping is performed. In this case, it does not matter what the second value is (**fail** or **ignore**).

Default = **off fail**

suffix *dn_suffix*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X	X			

Denotes the distinguished name of the root of a subtree in the namespace that is managed by this backend within the LDAP server. This option might be specified more than once to indicate all the roots of the subtrees within this backend except for the SDBM backend. The SDBM backend must have only one suffix. Note a suffix cannot be specified for the GDBM, CDBM, and EXOP backends. When the GDBM backend is configured, the **cn=changelog** suffix is reserved. When the CDBM backend is configured, the **cn=configuration** and **cn=ibmpolicies** suffixes are reserved. The special

suffix, **cn=localhost**, can be placed in any TDBM or LDBM backend and is exempt from replication when advanced replication is used.

Identical and overlapping suffixes cannot be specified in the LDAP server configuration file, even if the suffixes are within different backends. 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
```

See "Specifying a value for a distinguished name" on page 86 for information about specifying special characters and restrictions on attributes in the suffix.

Domain Component naming as specified in RFC 2247: *Using Domains in LDAP/X.500 Distinguished Names* is also supported in the LDAP server. For example, the domain name `ibm.com` could be specified as the following suffix in the configuration file:

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

supportKrb5 {on | off}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	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 servers root DSE.

Default = **off**

tcpTerminate {terminate | recover}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies whether the LDAP server ends when network interfaces are not active. The LDAP server periodically polls the network interfaces it is using to determine when they go down and come back up. If an interface fails but the LDAP server still has at least one active interface, the server continues processing and reestablishes a failed interface when it detects that it has become active. If all interfaces fail and **tcpTerminate terminate** is specified, the LDAP server ends. If **tcpTerminate recover** is specified, then the LDAP server remains active and attempts to reestablish network interfaces when it detects they have become active. All client operations targeted to the LDAP server fail until a network interface can be reconnected. The frequency of polling can be set using the **LDAP_NETWORK_POLL** environment variable. See "Environment variables used by the LDAP server" on page 179 for more information.

The **tcpTerminate** option is also used to determine whether the LDAP server ends if SSL or Kerberos initialization fails during server initialization. If **terminate** is specified, the LDAP server ends. If **recover** is specified, the LDAP server continues initialization, but the failed interface (SSL or Kerberos) cannot be used until the error is fixed and the LDAP server is restarted.

Default = **recover**

Note:
When the server is started as a transition server (started in transition mode), this configuration option is ignored and the server ends if network interface inactive is detected before transition completes. Once transition completes, the setting in the configuration file takes effect. See "Updating LDAP configurations settings in a sysplex without server outage" on page 210.

timeLimit *num-seconds*

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X	X	X	X	X	X	

Specifies the maximum number of seconds (in real time) the LDAP server spends answering a search 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 or in group search limits. Specifying this before a **database** line in the configuration file 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. Specifying a time limit using group search limits sets the limit only for the members of that group. See "Managing group search limits" on page 423 for more information about group search limits.

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 when determining the time limit for a search request.

- If the client has not bound as an administrator:
  - If a group search time limit exists for the requester, then the time used to limit the search is the smaller of the time limit passed by the client and the group search time limit. If the client does not specify a time limit on the search, then the group search time limit is used.
  - If a group search time limit does not exist for the requester, then the time that is used to limit the search is the smaller of the time limit that is passed by the client and the time limit that is determined by the server from the **timeLimit** configuration options in the configuration file. If the client does not specify a time limit, then the server time limit is used.
- If the client has bound as an administrator, the time limit is the value passed by the client. If the client does not specify a limit, then the amount of time is unlimited. The time limits from the configuration file and from group search limits are ignored when the client has bound as an administrator.

When accessing the z/OS LDAP server 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** option in the configuration file

(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.

There are additional considerations for time limit when performing a subtree search from the root DSE (a NULL-based search). See "Root DSE search with subtree scope (Null-based subtree search)" on page 596 for more information.

useAdvancedReplication {on | off}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
					X	

Specifies if the LDAP server supports advanced replication. If advanced replication is active, then the **masterServer, masterServerDN, masterServerPW, peerServer, peerServerDN**, and **peerServerPW** configuration options cannot be specified in any LDBM, TDBM, or CDBM backends.

Note:

- The LDAP server does not start when **useAdvancedReplication on** is specified and entries with an objectclass of **replicaObject** are present in a TDBM, LDBM, or CDBM backend. If entries with an objectclass of **replicaObject** are attempted to be added or modified in this configuration, the add or modify request is rejected.
- The LDAP server does not start when **useAdvancedReplication off** is specified and entries with an objectclass of **ibm-replicationAgreement, ibm-replicationContext, ibm-replicationGroup**, or **ibm-replicationSubEntry** are present in a TDBM, LDBM, or CDBM backend. If entries with these objectclass values are attempted to be added or modified in this configuration, the add or modify request is rejected.

See Chapter 26, "Advanced replication," on page 487 for additional information about advanced replication.

The server compatibility level must be at least 5 when **useAdvancedReplication on** is specified. See the **serverCompatLevel** configuration option at "serverCompatLevel {3 | 4 | 5 | 6 | 7| 8}" on page 125 for more information about the server compatibility level.

Default = **off**

useNativeAuth {selected | all | off}

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
	X	X			X	

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

- **selected**, only entries with the **ibm-nativeId** attribute that are within the native subtrees (see **nativeAuthSubtree** option at "nativeAuthSubtree {all | dn}" on page 114) 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.

Note: z/OS LDAP password policy does not apply to entries participating in native authentication.

Default = **off**

validateincomingV2strings {on | off }

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies whether the incoming strings are validated. If set to **on**, this setting limits the format of incoming string data that is 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 that is received on operations outside of the IA5 character set causes the operations to fail with **LDAP_PROTOCOL_ERROR**.

Default = **on**

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

wlmExcept *name* **[***IP_address***] [***dn***]**

Global	TDBM	LDBM	SDBM	GDBM	CDBM	EXOP
X						

Specifies the Workload Manager (WLM) transaction name that is used for client requests originating from an IP address or a bound user's distinguished name (DN). The **wlmExecpt** configuration option can be specified multiple times to allow the routing of different LDAP client requests to the same or different WLM transaction names. The order of the **wlmExcept** configuration options in the LDAP server configuration file determines the order the LDAP server uses to match incoming client requests and route them to the WLM transaction name. During LDAP server initialization, a WLM enclave is created for each unique name. See "Workload manager (WLM)" on page 602 for more information about configuring the LDAP server to use WLM.

name Specifies the WLM transaction name that is used for this enclave. The name must be 1-8 characters long and can consist of letters, numbers, and the special characters '$', '#', or '@'. The WLM transaction name must be configured in WLM. Multiple **wlmExcept** configuration options with the same name use the same enclave.

IP_address
Specifies the client's IPv4 or IPv6 address to be associated with this WLM enclave.

dn Specifies the bind user's distinguished name to be associated with this WLM enclave. For information about specifying a value for a distinguished name for this option, see "Specifying a value for a distinguished name" on page 86.

Note:

1. If both the *IP_address* and *dn* values are not specified with the **wlmExcept** configuration option, a WLM enclave is created with the

transaction value *name*. However, the enclave is not associated with any client requests until a WLMEXCEPT modify command is issued.

2. If both *IP_address* and *dn* are specified, only incoming client requests originating from that *IP_address* and bound as the *dn* are routed to the WLM transaction name specified.

Default = **GENERAL**

By default, the WLM transaction name, **GENERAL**, is used by the LDAP server for client requests originating from IP addresses or bind distinguished names that are not specified on **wlmExcept** configuration options. WLM transaction name **GENERAL** must be configured in WLM. See *z/OS MVS Planning: Workload Management* for more information about configuring WLM.

Deprecated options

The **database** option deprecates the use of the database type **exop**.

The **listen** option 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** option takes precedence over what was specified for the deprecated **security**, **port**, and **securePort** options. If using an earlier version of the configuration file that contains the **security**, **port**, or **securePort** options, the LDAP server is configured to listen on the port numbers that are specified for **securePort**, **port**, or both, depending upon the **security** setting. However, you might want the LDAP server to be configured using the **listen** option. See the description of the **listen** option at "listen ldap_URL" on page 104 for more information.

Ignored options

The **replKeyRingFile** and **replKeyRingPW** options are no longer necessary or evaluated by the LDAP server. These options are removed from the configuration file. Use the **sslKeyRingFile** configuration option to specify the key database file, RACF key ring, or PKCS #11 token. The **sslKeyRingPWStashFile** configuration option is used to specify the password stash file for the key database file while the **sslKeyRingFilePW** configuration option is used to specify the password of the key database file.

The **maxThreads** and **waitingThreads** options are no longer necessary or evaluated by the LDAP server. These options are also 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 at "commThreads num-threads" on page 96 for more information.

The **databasename** and **verifySchema** options are no longer necessary or evaluated by the LDAP server. These options are removed from the configuration file.

The **sysplexGroupName** and **sysplexServerName** options are no longer necessary or evaluated by the LDAP server. These options are removed from the configuration file. Use the **serverSysplexGroup** option to identify the cross-system coupling facility (XCF) group.

CDBM backend configuration and policy entries

When the CDBM backend is configured in the LDAP server configuration file, configuration related entries are stored in the **cn=configuration** suffix while policy entries are stored in the **cn=ibmpolicies** suffix. These entries contain attributes that represent configuration options. The attribute values can be changed dynamically by an LDAP modify command while the LDAP server is running. All changes take effect immediately, without needing to restart the server. By default, the CDBM backend only allows an LDAP root administrator to modify the configuration and policy entries, but access can be changed by modifying the ACL on these entries. Some administrative roles allow modifying configuration and policy entries in the CDBM backend without modifying ACLs. See Chapter 9, "Administrative group and roles," on page 159 for more information about administrative roles authority.

When the LDAP server starts, the configuration and policy entries that do not exist are created with each attribute assigned to its default value. If an attribute value is deleted, the default value is used. The deleting and renaming of advanced replication configuration entries is only supported when **useAdvancedReplication off** is specified in the CDBM backend. The deleting of the **cn=pwdpolicy,cn=ibmpolicies** entry is only supported when the server compatibility is set to less than 6.

This section contains information about the entries that exist under the **cn=configuration** and **cn=ibmpolicies** suffixes and the attribute values in these entries that affect the configuration of the LDAP server.

cn=configuration

This is a container entry that is used to define dynamic configuration attributes. Table 10 describes the entry attribute descriptions.

Table 10. **cn=configuration** *entry attribute descriptions*

Attribute description and default
cn Specifies the common name of the configuration entry. This attribute is never interpreted by the server. **Default:** Configuration
ibm-slapdSAFSecurityDomain Specifies the high-level component of the resource profile names that are used to define LDAP-related information in the z/OS security manager. This high-level qualifier is used to define administrative roles in the z/OS security manager. The length of this value cannot be more than 228 characters and cannot contain a blank, comma, parenthesis, semicolon, asterisk, percent sign, or ampersand. See Chapter 9, "Administrative group and roles," on page 159 for more information about configuring administrative roles in the z/OS security manager. **Default:** GLDSEC
ibm-slapdAdminGroupEnabled A boolean (true or false) used to specify if the administrative group is currently enabled. If set to true and the server compatibility level is 7 or greater, the administrative group is enabled and the LDAP root administrator can delegate server administration authority. If set to false, the administrative group cannot be enabled in the LDAP server. **Default:** false **Note:** This is a user-modifiable operational attribute and is only returned on a search request when specifically listed in the list of return attribute types or a '+' is specified in the list of returned attribute types.

Table 10. **cn=configuration** *entry attribute descriptions (continued)*

Attribute description and default
ibm-slapdPagedResAllowNonAdmin A boolean (true or false) used to indicate whether the server allows non-administrators to request paged search results. If set to true, the server accepts any paged search request, including those submitted by a user binding anonymously. If set to false, the server only accepts paged search requests submitted by a user with administrator authority. In this case, the criticality of the **pagedResults** server control determines how the server handles a paged search request from non-administrator users. If the control is specified as critical, the request is rejected with an **LDAP_INSUFFICIENT_ACCESS** return code. If the control is specified as non-critical, the search is performed, but all results are returned without paging. The **ibm-slapdPagedResLmt** attribute must be set to a value greater than zero to enable paged search results. **Default:** false **Note:** This is a user-modifiable operational attribute and is only returned on a search request when specifically listed in the list of return attribute types or a '**+**' is specified in the list of returned attribute types.
ibm-slapdPagedResLmt Specifies the maximum number of outstanding paged search requests allowed simultaneously on a single connection. If the maximum number of outstanding paged search requests is exceeded, the criticality of the **pagedResults** server control determines how the server handles the paged search request. If the control is specified as critical, the request is rejected with an **LDAP_ADMIN_LIMIT_EXCEEDED** return code. If the control is specified as non-critical, the search is performed but all results are returned without paging. The value must be between 0 and the maximum integer size. A value of 0 indicates that paged search results are not supported. **Default:** 0 **Note:** This is a user-modifiable operational attribute and is only returned on a search request when specifically listed in the list of return attribute types or a '**+**' is specified in the list of returned attribute types.
ibm-slapdServerID Specifies a short descriptive name of this server in an advanced replication environment. This server name is used when configuring the relationships among LDAP servers in an advanced replication environment and therefore a unique **ibm-slapdServerID** value is chosen for each server in the replication topology. This value is displayed as the **ibm-serverID** attribute value in the root DSE entry. The **ibm-slapdServerID** value is used in the replica subentry attribute **ibm-replicaServerID**, therefore, this value cannot be modified once replica subentries are created in the directory. This value cannot be deleted if advanced replication is configured. **Default:** A randomly generated attribute value like an **ibm-entryUUID** attribute value that is created when the CDBM backend is first initialized.

Table 10. **cn=configuration** *entry attribute descriptions (continued)*

Attribute description and default
ibm-slapdSortKeyLimit Specifies the maximum number of sort keys that can be included on a single sorted search request. If the maximum number of sort keys is exceeded, the criticality of the **SortKeyRequest** server control determines how the server handles the sorted search request. If the control is specified as critical, the request is rejected with an **LDAP_UNAVAILABLE_CRITICAL_EXTENSION** return code. If the control is specified as non-critical, the search is performed, but unsorted results are returned and an **LDAP_ADMIN_LIMIT_EXCEEDED** sort result code is returned in the **SortKeyResponse** server control. The value must be between 0 and the maximum integer size. A value of 0 indicates that sorted search results are not supported. **Default:** 0 **Note:** This is a user-modifiable operational attribute and is only returned on a search request when specifically listed in the list of return attribute types or a '**+**' is specified in the list of returned attribute types.
ibm-slapdSortSrchAllowNonAdmin A boolean (true or false) used to indicate whether the server allows non-administrators to request sorted search results. If set to true, the server accepts any sorted search request, including those submitted by a user binding anonymously. If set to false, the server only processes sorted search requests submitted by a user with administrator authority. In this case, the criticality of the **SortKeyRequest** server control determines how the server handles a sorted search request from non-administrator users. If the control is specified as critical, the request is rejected with an **LDAP_UNAVAILABLE_CRITICAL_EXTENSION** return code. If the control is specified as non-critical, the search is performed but unsorted results are returned and an **LDAP_INSUFFICIENT_ACCESS** sort result code is returned in the **SortKeyResponse** server control. The **ibm-slapdSortKeyLimit** attribute must be set to a value greater than zero to enable sorted search results. **Default:** false **Note:** This is a user-modifiable operational attribute and is only returned on a search request when specifically listed in the list of return attribute types or a '**+**' is specified in the list of returned attribute types.

cn=Replication,cn=configuration

This entry is used to configure many aspects of advanced replication such as the maximum number of pending or failed replication changes displayed for a replication agreement. See Chapter 26, "Advanced replication," on page 487 for more information.

Table 11. **cn=Replication,cn=configuration** *entry attribute descriptions*

Attribute description and default
cn Specifies the common name of the configuration entry. This attribute does not affect advanced replication configuration. **Default:** Replication
ibm-replicationOnHold A boolean (true or false) used to indicate whether replication is suspended from all replication agreements in the server. If set to true, replication from all replication agreements is suspended and updates are queued. If set to false, replication updates are handled normally by each replication agreement. **Default:** false

Table 11. **cn=Replication,cn=configuration** *entry attribute descriptions (continued)*

Attribute description and default
ibm-slapdMaxPendingChangesDisplayed Specifies the maximum number of pending replication changes and the maximum number of failed replication changes that are displayed when searching a replication agreement on a supplier server. Increase this value if more pending and failed changes must be displayed for each replication agreement. The pending replication changes are stored in the replication agreement entry in the **ibm-replicationPendingChanges** multi-valued operational attribute. The failed replication changes are stored in the **ibm-replicationFailedChanges** multi-valued operational attribute. See Table 82 on page 544 for more information about these operational attributes. The value must be between 0 and the maximum integer size. A value of 0 indicates that no pending changes are displayed for each replication agreement. **Default:** 200
ibm-slapdReplConflictMaxEntrySize Specifies the maximum length (in bytes) for all attribute values in an entry for replication conflict resolution to occur. If a replication conflict occurs on the consumer server and the total attribute value length for all values in an entry is less than or equal to this number, the entry is resent to the consumer server to automatically correct the replication conflict. Otherwise, the entry is not resent to the consumer server. This value applies to each replication agreement in the server. Increase this value when large entries are modified so that out of sync conditions between a supplier and consumer server can be resolved automatically with conflict resolution. If automatic replication conflict resolution support is not wanted, set this value to a small number. The value must be between 0 and the maximum integer size. A value of 0 indicates that all entries are resent to the consumer server regardless of the size of the entry. **Default:** 0
ibm-slapdReplContextCacheSize Specifies the maximum size of each advanced replication context cache, in bytes. An advanced replication context cache is used to store pending replication updates for each replication context in the server. This cache reduces the number of queries to the backends to find the same information. Increase the size of the cache when replicating more and larger entries, such as large group entries. This value must be between 0 and the maximum integer size. A value of 0 indicates that there are no replication context caches in the server. **Default:** 100000
ibm-slapdReplMaxErrors Specifies the maximum number of advanced replication failures that are logged for each backend in the server. If there are multiple replication agreement entries in a backend, each agreement shares the maximum number of replication failures allowed for the backend. The failed replication changes are stored in the replication agreement entry in the **ibm-replicationFailedChanges** multi-valued operational attribute. When the number of replication failures exceeds this value, advanced replication for this agreement stalls. See “Monitoring and diagnosing advanced replication problems” on page 543 for more information about recovering from out of sync and stall conditions. This value must be between -1 and the maximum integer size. A value of 0 indicates that advanced replication failures are not logged for any replication agreements, therefore, replication stalls at the first failed replication update. A value of -1 indicates that an unlimited number of advanced replication failures are logged for all replication agreements. **Default:** 0

Table 11. **cn=Replication,cn=configuration** *entry attribute descriptions (continued)*

Attribute description and default
ibm-slapdReplRestrictedAccess A boolean (true or false) used to control access to replication topology entries (replication contexts, groups, subentries, and agreements). This attribute provides a way to limit access to the replication topology entries in the LDAP server. If set to true, only LDAP root, directory data, or replication administrators, and the master server DN have access to replication topology entries. If set to false, non-administrator users must have the proper authority to access the replication topology entries. **Default:** false **Note:** This is a user-modifiable operational attribute and is only returned on a search request when listed in the list of return attribute types or a '**+**' is specified in the list of returned attribute types.
ibm-slapdReplicateSecurityAttributes Enables replication of security attributes between the read-only replica and master so that password policy for account lockout can be enforced in replication topologies. **Note:** This capability is only supported for bind operations that use simple authentication, and authentication that is done with compare operations that involve the **userPassword** attribute. **Default:** false

cn=Log Management,cn=Configuration

This is a container entry that does not contain any attribute values that affect the configuration of the LDAP server.

cn=Replication,cn=Log Management,cn=Configuration

This entry is used by advanced replication to specify the location of the lost and found log file. See Chapter 26, "Advanced replication," on page 487 for more information.

Table 12. **cn=Replication,cn=Log Management,cn=Configuration** *entry attribute descriptions*

Attribute description and default
cn Specifies the common name of the configuration entry. This attribute does not affect advanced replication configuration. **Default:** Replication
ibm-slapdLog Specifies the z/OS UNIX System Services file name and directory location for the lost and found log file. The lost and found log file is created by the consumer server the first time a replication conflict occurs. Any entries that are deleted because of a replication conflict are stored in LDIF format in this file. The directory path that is specified in this attribute value must exist before the file is created, otherwise replication conflicts are not written to the lost and found log file. This value cannot be deleted when advanced replication is configured. If this value is modified, the original value is still used until the LDAP server is restarted. **Default:** /var/ldap/logs/lostandfound.log

cn=admingroup,cn=configuration

This is a container entry that does not contain any attribute values that affect the configuration of the LDAP server.

cn=safadmingroup,cn=configuration

This entry is used to define administrative group members whose roles are defined in RACF. Group members are added to the entry by adding the optional member attribute to the entry and roles are defined in RACF. See Chapter 9, "Administrative group and roles," on page 159 for more information.

Table 13. **cn=safadmingroup,cn=configuration** *entry attribute descriptions*

Attribute description and default
cn A required attribute that specifies the common name of the SAF administrative group entry. This attribute does not affect administrative role processing. **Default:** safadmingroup
member An optional attribute that specifies a distinguished name (DN) leading to a SAF user ID. Examples of these DNs are: SDBM entries, DNs from an SSL client certificate, DN of a TDBM or LDBM entry participating in native authentication, and a Kerberos mapped DN. This DN is checked against each administrative role defined in the LDAP general resource class in RACF to see if the user has READ authority to the profile. If the user has READ access to the profile, the user is granted that administrative role. **Default:** none

cn=ibmpolicies

This is a container entry that does not contain any attribute values that affect the configuration of the LDAP server.

cn=pwdpolicy,cn=ibmpolicies

This entry is used to configure the global password policy. When the global password policy is activated, this policy applies to all entries that have passwords stored in the LDBM, TDBM, and CDBM backends unless there is an overriding individual or group password policy in effect. See Table 45 on page 367 for the attribute descriptions of the **cn=pwdpolicy,cn=ibmpolicies** entry and the default values in this entry.

Configuration considerations

The following table shows all 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 14. Configuration considerations

Dependency	More information
Operational mode You must determine the type of operational mode your LDAP server runs in. For example, single-server mode or multi-server mode.	"Determining operational mode" on page 146
TDBM backend You can use a TDBM backend database based on DB2.	"Setting up for TDBM" on page 59

Table 14. Configuration considerations (continued)

Dependency	More information
SDBM backend You can use an SDBM backend database based on RACF.	"Setting up for SDBM" on page 60
LDBM backend You can use an LDBM backend database based on a z/OS UNIX System Services file system.	"Setting up for LDBM" on page 61
GDBM backend You can use a GDBM backend database based on DB2 or based on a z/OS UNIX System Services file system to log changes to entries within the LDAP server and to RACF user, group, connection, and general resource profiles.	"Configuring file-based GDBM" on page 64
CDBM backend You can use a CDBM backend database based on a z/OS UNIX System Services file system for configuration information.	"Setting up for CDBM" on page 62
EXOP backend You can use an EXOP backend to retrieve Policy Director data. Use of the EXOP backend is deprecated.	Chapter 22, "Using extended operations to access Policy Director data," on page 417
Plug-in extension You can use a plug-in to extend the capabilities of the LDAP server.	*z/OS IBM Tivoli Directory Server Plug-in Reference for z/OS*
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 68
Encryption or hashing Your LDAP server can prevent unauthorized access to **ibm-slapdMasterPw, ibm-replicaKeyPwd, ibm-slapdAdminPw, userPassword, secretKey**, and **replicaCredential** values in a TDBM, LDBM, or CDBM backend database.	"Configuring for encryption or hashing" on page 77
Kerberos authentication You can enable GSS API Kerberos binds and configure identity mapping.	Chapter 19, "Kerberos authentication," on page 393
Native authentication You can enable and configure your directory to perform authentication using the Security Server.	Chapter 20, "Native authentication," on page 403
CRAM-MD5 and DIGEST-MD5 authentication The LDAP server can be configured to perform CRAM-MD5 and DIGEST-MD5 authentication binds.	Chapter 21, "CRAM-MD5 and DIGEST-MD5 authentication," on page 413

Table 14. Configuration considerations (continued)

Dependency	More information
Administrator DN and replica server DN and passwords Determine how to set up your root administrator in the configuration file and password. Also determine how to set up your master or peer server DN and password, if you are using basic replication.	"Establishing the root administrator DN and basic replication replica server DN and passwords" on page 151
Basic replication To keep multiple databases in sync, you can use basic replication.	Chapter 25, "Basic replication," on page 469 or "Establishing the root administrator DN and basic replication replica server DN and passwords" on page 151
Advanced replication The LDAP server can be configured to replicate only portions of a backend to another LDAP server. This keeps multiple databases on different LDAP servers in sync.	Chapter 26, "Advanced replication," on page 487
Referrals To refer clients to additional directory servers, use referrals.	Chapter 29, "Referrals," on page 575
Password policy Passwords stored in the LDBM, TDBM, and CDBM backends can be configured for password policy to control how passwords are used and administered.	Chapter 18, "Password policy," on page 365
Administrative group and roles An LDAP root administrator can delegate administrative functions by creating the administrative group and roles.	Chapter 9, "Administrative group and roles," on page 159

"Example configuration scenarios" on page 154 has various examples showing different LDAP server configurations.

Determining operational mode

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

- **Single-server mode**

  In this operational mode, only a single instance of the LDAP server might use a given TDBM, LDBM, CDBM, or GDBM database to store directory data. This server might perform basic or advanced replication (see Chapter 25, "Basic replication," on page 469 or Chapter 26, "Advanced replication," on page 487) of TDBM, LDBM, or CDBM database changes to other servers (on the same host system or on another host system).

  See "Operating in single-server mode" on page 148 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, LDBM, CDBM, or GDBM backends. This server might perform basic or advanced replication (see Chapter 25, "Basic replication," on page 469 or Chapter 26, "Advanced

replication," on page 487) of TDBM, LDBM, or CDBM database changes to other servers (on the same host system or on another host system). However, each server must have its own separate set of replica servers.

See "Operating in multiple single-server mode" on page 148 for more information.

- **Multi-server mode**

  In this operational mode, multiple concurrent instances of the LDAP server use the same TDBM, LDBM, CDBM, or GDBM database to store directory data. The LDAP servers might run on the same host system or on different host systems. In both cases, XCF messaging (Parallel sysplex support) is used to communicate between the LDAP servers. 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, and dynamic workload balancing across constituent systems in the sysplex. For additional information, see Parallel Sysplex Overview (http://www.ibm.com/systems/z/advantages/pso/sysover.html).

  These servers might perform basic or advanced replication of TDBM, LDBM, or CDBM database changes to other servers (on the same host system or on another host system). However, each server must have the same set of replica servers. See Chapter 25, "Basic replication," on page 469 or Chapter 26, "Advanced replication," on page 487 for more information.

  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 that are functionally equivalent and that provide access to the same LDAP directory data.

  See "Operating in multi-server mode" on page 149 for more information.

After the operational mode is determined for your site, the server compatibility features that are enabled in the z/OS LDAP server is determined by the setting of the **serverCompatLevel** configuration option in the LDAP server configuration file. The server compatibility setting is especially important when running in multiserver mode when sharing backends among LDAP servers running on different releases. In these environments, the **serverCompatLevel** option must be set to the earliest z/OS LDAP server release that is shared. See "serverCompatLevel {3 | 4 | 5 | 6 | 7| 8}" on page 125 for more information about server compatibility and the supported features at each compatibility level.

If there are DB2-based (TDBM and GDBM) backends in the LDAP server configuration file, the **serverCompatLevel** configuration option or its default value sets the DB_VERSION value in the DIR_MISC table. If there is a difference between the DB_VERSION and **serverCompatLevel** values when the LDAP server is started, the DB_VERSION value is set to the **serverCompatLevel** configuration option value. Therefore, it is important to set the **serverCompatLevel** configuration option appropriately when sharing DB2-based backends among LDAP servers running on different releases.

The following operational mode can be used with any of the modes described above.

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

  The program call (PC) callable support in LDAP directory provides a program call interface to the LDAP directory change log backend (GDBM). This interface

is only available using the z/OS SAF interfaces designed to allow RACF to log changes to RACF data in the LDAP change log.

See "Operating in PC callable support mode" on page 151 for more information.

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

Note:

1. A single LDAP server instance can have one SDBM backend, one GDBM backend, and one CDBM backend, but it can have multiple TDBM and LDBM 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 is accessed from both servers, then the RACF database is also shared across the systems where the LDAP servers run to ensure consistency of SDBM operations. See *z/OS Security Server RACF System Programmer's Guide* for information about setting up a shared RACF database.

Operating in single-server mode

For the LDAP server to operate in single-server mode, the server configuration file might contain any of the previously documented options except the **serverSysplexGroup** option (the presence that causes the LDAP server to operate in multi-server mode). If the **multiserver** option is present, its value must be set to **off**.

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 might yield unpredictable results if one or more of those server instances is configured in single-server mode. If you want 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 multiple single-server mode

For the LDAP server to operate in multiple single-server mode, there must be two or more LDAP servers running on the same system, each running in single-server mode. The server configuration file might contain any of the previously documented options except the **serverSysplexGroup** option (the presence that causes the LDAP server to operate in multi-server mode). If the **multiserver** option is present, its value must be set to **off**.

Restrictions

The LDAP servers cannot share TDBM, LDBM, CDBM, or GDBM backends and cannot share the LDAP server schema. This means that each server must have unique values for the **databaseDirectory** configuration option (for an LDBM, file-based GDBM, or CDBM backend), the **dbuserid** configuration option (for a TDBM or DB2-based GDBM backend), and the **schemaPath** configuration option (for the schema).

Setting up multiple LDAP servers with DB2-based backends

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 55 and be sure to:
  1. Modify the SPUFI file that creates tables and indexes. Change `-DB_USERID-` (database owner) and `-DB_NAME-` (database name) to a different value and submit the SPUFI. A separate set of DB2 tables is created.
  2. Update the **dbuserid** configuration option with the `-DB_USERID-` value from step 1.

Operating in multi-server mode

For the LDAP server to operate in multi-server mode, the global section of the server configuration file must contain the **serverSysplexGroup** option. This option specifies the name of the cross-system coupling facility (XCF) group and activates the LDAP sysplex support. All the LDAP servers in the sysplex that specify the same group name share the LDAP server schema and the directories of backends that specify **multiserver on** in their backend section of the server configuration file. Each LDAP server in the XCF group must specify the same value for the **schemaPath** option and must have read/write access to the specified directory or to `/var/ldap/schema` if the option is not specified. The schema directory that is used must exist within a shared z/OS UNIX System Services file system and must be accessible to all servers in the XCF group. An LDBM, CDBM, or file-based GDBM backend is shared when **multiserver on** is specified in the backend section. When an LDBM, CDBM, or file-based GDBM backend is shared, the **databaseDirectory** configuration option must have the same value, exist within a shared z/OS UNIX System Services file system, and must be accessible to all servers in the XCF group. Each server must have read/write access to the specified directory or to `/var/ldap/ldbm` (LDBM default), `/var/ldap/schema` (CDBM default if **schemaPath** is not specified), `/var/ldap/gdbm` (GDBM default) if the **databaseDirectory** configuration option is not specified. See *z/OS UNIX System Services Planning* for information about setting up a shared z/OS UNIX System Services file system.

The port specified in the **listen, port**, and **securePort** options must be the same on each LDAP server in the sysplex group for dynamic workload balancing to function properly in the sysplex.

When you are using referrals, 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 that point to the multiple server instances can have multi-valued **ref** attributes set up, each of which is an LDAP directory URL pointing to the corresponding server instances.

When LDAP servers in a sysplex group are started, the first server to start becomes the sysplex owner of all the shared backends, including the schema. The LDAP server **DISPLAY XCF** operator modify command can be used to determine the active servers in the sysplex group and shows which one is the owner. See "Displaying performance information and server settings" on page 184 for more information.

The sysplex owner is responsible for making all updates to the schema and to any shared LDBM or CDBM backends. If changes to the schema, LDBM, or CDBM entry are directed to another server in the sysplex group, that server uses XCF to forward the change to the sysplex owner. The sysplex owner makes the change,

both in memory and in the database checkpoint file, and then uses XCF to broadcast the change to the other servers. The other servers update their directory in memory.

For a shared TDBM backend, any server in the sysplex group can update a TDBM entry in the database in DB2. That server then notifies all the other servers (including the sysplex owner) by way of XCF that a change has taken place. These servers refresh their various caches so that out-of-date contents are not used. This sysplex support for caches is only available when the **serverCompatLevel** configuration option is set to **4** or higher. If the **serverCompatLevel** configuration option is not at this level or higher, the servers in the sysplex do not support caching in the shared TDBM backend.

All replication is handled by the sysplex owner. If a shared backend is replicated, the sysplex owner creates the necessary replication information for a change to an entry in that backend and sends it to the replica or consumer servers.

If the GDBM backend is shared, GDBM is contacted to create a change log record on the server that makes the change. GDBM processing depends on how GDBM is configured. If GDBM is file-based, it handles the change log request like LDBM: the change log request is forwarded to the sysplex owner, who creates the change log record in GDBM and notifies the other servers. When GDBM is DB2-based, it handles the request such as TDBM: the DB2 database is updated locally and the other servers are notified.

If the LDAP server that is the sysplex owner ends, another LDAP server in the sysplex group becomes the owner.

Dependencies

When sharing a DB2-based backend on multiple systems in a sysplex, the DB2 subsystems that each server instance attaches to must be configured on each of the images as members of the same DB2 data sharing group. See IBM Information Management Software for z/OS Solutions Information Center for information about planning, installing, and setting up DB2 for data sharing, CLI and ODBC. Also, see *z/OS MVS Setting Up a Sysplex* for information about planning and installing a Parallel Sysplex.

Restrictions

Each LDAP server in the XCF group must specify the same value for the **schemaPath** configuration option and must have read/write access to the specified directory or to `/var/ldap/schema` if the option is not specified. The schema directory that is used must exist within a shared z/OS UNIX System Services file system and must be accessible to all servers in the XCF group. See *z/OS UNIX System Services Planning* for information about setting up a shared z/OS UNIX System Services file system.

You can configure one backend to be shared (**multiserver on**) while another backend is not shared (**multiserver off**) except when GDBM or CDBM is configured. When GDBM or CDBM is configured, all TDBM, LDBM, CDBM, and GDBM backends must be configured to be shared or all must be not shared.

Note: The following restrictions apply to the backend section of the configuration file on each LDAP server that is sharing the backend:

- The *name* parameter must be specified on the **database** option and must have the same value on each server.
- The values of the **suffix** option must be the same on each server.

- For a DB2-based backend, the **dbuserid** option must have the same value.
- For a file-based backend, the **databaseDirectory** configuration option must have the same value, exist within a shared z/OS UNIX System Services file system, and must be accessible to all servers in the XCF group. Each server must have read/write access to the specified directory or to `/var/ldap/ldbm` (LDBM default), `/var/ldap/schema` (CDBM default if the **schemaPath** configuration option is also not specified), `/var/ldap/gdbm` (GDBM default) if the **databaseDirectory** configuration option is not specified. See *z/OS UNIX System Services Planning* for information about setting up a shared z/OS UNIX System Services file system.
- The SDBM backend does not support the **multiserver** configuration option. RACF provides the sysplex sharing support for the RACF database. See *z/OS Security Server RACF System Programmer's Guide* for information about setting up a shared RACF database.

Operating in PC callable support mode

The program call (PC) callable support in the LDAP server provides a program call interface to the LDAP server change log backend (GDBM). This interface is only available using the z/OS SAF interfaces designed to allow RACF to log changes to RACF data in the LDAP server 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 typical 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 reusable 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 restarted.

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 the LDAP server in a sysplex, configure one LDAP server on each system in the sysplex to run the PC callable support. Each system shares the GDBM, CDBM, and RACF databases to ensure that they return the same results.

Establishing the root administrator DN and basic replication replica server DN and passwords

There are several ways that the LDAP root administrator and replica server distinguished names and password values can be configured. One of these ways must be used because an LDAP root administrator DN and password are required for the LDAP server and some other LDAP directory programs to operate:

- A root administrator DN must be specified in the global section of the configuration file using the **adminDN** configuration option (see "Configuration file options" on page 90). The password for this root administrator DN can optionally (this is not suggested) be placed in the configuration file using the **adminPW** configuration option (see "Configuration file options" on page 90) or can be held in the namespace managed by this instance of the LDAP server.
- If a replica is being established for a backend, the **masterServerDN** or **peerServerDN** configuration option must be specified in the backend section of the configuration file. The **masterServerPW** or **peerServerPW** configuration option can optionally be specified. (This is also not suggested.) All the options described below are applicable for **adminDN** and the first three options described below are applicable for **masterServerDN** and **peerServerDN**.

Note: The LDAP root administrator can delegate server administrator responsibility by defining the administrative group, adding members, and assigning administrative roles. Administrative roles can be defined in LDAP by an LDAP root administrator or in RACF by a RACF administrator. See Chapter 9, "Administrative group and roles," on page 159 for more information.

- Root administrator DN and password in configuration file

  The simplest but least secure method is to select a root administrator DN that is outside of the scope of suffixes managed by this server (see the **suffix** configuration option, "Configuration file options" on page 90). In other words, choose a root 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 root administrator DN requires that the password is placed in the configuration file using the **adminPW** configuration option (see "Configuration file options" on page 90). There is no password policy support for the root administrator DN when the password is defined in the configuration file.

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

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

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

  When a program or user binds using this root 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, LDBM, or CDBM backend, it might be necessary to use this approach until the schema supporting the directory entries is loaded. When the schema is loaded and the entry representing the root administrator is added, the **adminDN** can be changed to the entry DN (see the next list item regarding "Root administrator DN and password as a TDBM, LDBM, or CDBM entry"). The server must be restarted to pick up the new **adminDN**.

- Root administrator DN and password as a TDBM, LDBM, or CDBM entry

  In this method, the root administrator DN is established as an entry managed by a TDBM, LDBM, or CDBM backend. The **userPassword** attribute is used to hold the password for the root administrator DN in this case. There is LDAP password policy support for the root administrator DN when the entry contains a **userPassword** attribute value in the LDBM, TDBM, or CDBM backend.

Alternatively, the password or password phrase can be stored in the z/OS security manager if native authentication is configured. The LDAP password policy support does not apply to the root administrator DN when the password or password phrase exists in the z/OS security manager. The z/OS security manager provides the authentication rules and enforcement of its own password policy. See Chapter 20, "Native authentication," on page 403 for more information about using native authentication.

For example, if the TDBM or LDBM backend is managing the portion of the namespace `"o=Your Company"`, one root administrator DN that could be selected is `"cn=LDAP Admin,o=Your Company"`.

The configuration file includes the following options:

```
adminDN "cn=LDAP Admin,o=Your Company"
...
database tdbm GLDBTD31/GLDBTD64
...
suffix "o=Your Company"
```

The LDIF-format entry to be added to the database through **ldapadd** or **ldif2ds** 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, and the actual distinguished name.

If this entry is contained in a file system file called `admin.ldif`, it can be loaded using **ldapadd**:

```
ldapadd -h ldaphost -p ldapport -D binddn -w passwd -f admin.ldif
```

Note: The **ldapadd** example above assumes that the LDAP server is running and that the suffix entry (entry with the name **"o=Your Company"**) 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 root administrator DN 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.

For a TDBM backend, you can use the load utility, **ldif2ds**, to load the root administrator DN entry.

When a program or user binds using this root administrator DN, the LDAP server verifies that the password supplied on the request matches the value of the **userPassword** attribute stored in the entry.

CRAM-MD5 and DIGEST-MD5 authentication binds with the **adminDN** are supported if the entry exists in a TDBM, LDBM, or CDBM backend. The **adminDN** entry must contain a **uid** attribute value that is used as the user name by a client application when attempting a CRAM-MD5 or DIGEST-MD5 authentication bind. See Chapter 21, "CRAM-MD5 and DIGEST-MD5 authentication," on page 413 for more information.

- Root administrator DN and password in RACF

This method requires that the LDAP server is configured to use the RACF support provided in the SDBM backend. The root administrator DN can be established as a RACF-style DN based upon a RACF user ID. (See "RACF-style distinguished names" on page 273 for more information.) In this case, the password for the root administrator DN is the RACF user IDs password or password phrase, and is stored and verified by RACF. The LDAP password policy support does not apply to the root administrator DN when the password or password phrase is stored in RACF. RACF provides the authentication rules and enforcement of its own password policy.

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 includes these options:

```
adminDN "racfid=g1admin,profiletype=user,sysplex=Sysplex1,o=Your Company"
...
database sdbm GLDBSD31/GLDBSD64
suffix "sysplex=Sysplex1,o=Your Company"
```

When a program or user binds using this root administrator DN, the LDAP server makes a request to RACF to verify that the password supplied on the request matches the RACF password or password phrase for RACF user ID `g1admin`.

Note the RACF user ID specified must have an OMVS segment defined and an OMVS UID present.

- **krbLDAPAdmin** option in the configuration file

  You might want to configure the root administrator to be able to bind to the server through Kerberos. In this case, you must create a Kerberos identity for the user and also add this value to the configuration file. You cannot use the **krbLDAPAdmin** configuration option to provide a Kerberos identity for the **masterServerDN** or **peerServerDN** option. There is no LDAP password policy support for the Kerberos LDAP root administrator because there is not a **userPassword** attribute value stored in an LDBM, TDBM, or CDBM backend for the Kerberos identity.

  For example, if the RACF user ID associated with the LDAP root administrator is **LDAPADM** and this identity is configured in Kerberos to be `ldapadm@realm1.com`, then your configuration file contains 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. Only some of the options that can be specified for each section of the LDAP server configuration file are shown. See Table 9 on page 88 for a complete list of the options that are available for each section.

Configuring a TDBM backend with SSL/TLS and password encryption or hashing

The configuration example in this section uses the TDBM backend and shows a sample configuration file.

Sample **ds.conf** for TDBM, SSL/TLS, and password encryption or hashing:

```
# Filename ds.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 GLDBTD31/GLDBTD64 LocalDirectory
suffix "o=Your Company"
servername LOC1
dbuserid GLDSRV
attrOverflowSize 500
pwEncryption MD5
```

Configuring SDBM and GDBM (DB2-based) backends

The configuration example in this section uses SDBM and GDBM backends and shows a sample configuration file. In this example, the GDBM backend is based on DB2.

Sample **ds.conf** for SDBM and GDBM:

```
# Filename ds.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 GLDBSD31/GLDBSD64
suffix "cn=myRACF"
enableResources on

# GDBM backend section
database gdbm GLDBGD31/GLDBTD64
servername LOC1
dbuserid GLDSRV
attrOverflowSize 500
```

Configuring SDBM and TDBM backends

The configuration example in this section uses both SDBM and TDBM backends and shows a sample configuration file.

Sample **ds.conf** for SDBM and TDBM:

```
# Filename ds.conf

# Global section
sizelimit 500
timelimit 3600
adminDn "racfid=ldadmin,profiletype=user,cn=myRACF"
listen ldap://:389

# SDBM backend section
database sdbm GLDBSD31/GLDBSD64
```

```
suffix "cn=myRACF"
enableResources on

# TDBM backend section
database tdbm GLDBTD31/GLDBTD64
suffix "o=Your Company"
servername LOC1
dbuserid GLDSRV
attrOverflowSize 500
```

Configuring LDBM with native authentication and GDBM (file-based) backends

The configuration example in this section uses both LDBM and GDBM backends and shows a sample configuration file.

The GDBM backend is based on the z/OS UNIX System Services file system.

Sample **ds.conf** for LDBM and GDBM:

```
# Filename ds.conf

# Global section
sizelimit 500
timelimit 3600
adminDn "cn=LDAP Administrator,o=My Company
listen ldap://:389

# GDBM backend section
database gdbm GLDBGD31/GLDBGD64

# LDBM backend section
database ldbm GLDBLD31/GLDBLD64
suffix "o=My Company"
usenativauth all
nativeauthsubtree all
```

Configuring LDBM and CDBM backends with advanced replication and password policy

The configuration example in this section uses both CDBM and LDBM backends and shows a sample configuration file. Password policy is supported because CDBM is configured and the server compatibility level is 6 or higher (by default).

Sample **ds.conf** for CDBM and LDBM:

```
# Filename ds.conf

# Global section
sizelimit 500
timelimit 3600
adminDn "cn=LDAP Administrator,o=My Company
listen ldap://:389

# LDBM backend section
database ldbm GLDBLD31/GLDBLD64
suffix "o=My Company"

# CDBM backend section
database cdbm GLDBCD31/GLDBCD64
useAdvancedReplication on
```

Configuring an EXOP backend

The configuration example in this section uses an EXOP backend and shows a sample configuration file. Use of the EXOP backend is deprecated.

Sample **ds.conf** for EXOP:

```
# Filename ds.conf

# Global section
listen ldap://:pc

# EXOP backend section
database exop GLDXPD31/GLDXPD64
```

Chapter 9. Administrative group and roles

The administrative group enables 24 hour administrative capabilities without needing to share a single user ID and password among the administrators. It is a way for the LDAP root administrator defined in the configuration file to delegate a limited set of administrative tasks to one or more individual user accounts. Administrative group members are users that are added to the administrative group and assigned one or more administrative roles. These roles define the tasks that a group member is authorized to perform. The roles are assigned in the LDAP administrative group member entry or alternately the roles are assigned in RACF.

The administrative roles supported in the z/OS LDAP server are:

- **Directory data administrator** is allowed to administer all TDBM and LDBM backend entries and entries that exist under the **cn=ibmpolicies** suffix in the CDBM backend.
- **No administrator** is intended to quickly revoke the administrative group member's administrative privileges.
- **Operational administrator** is allowed to specify the **PersistentSearch** control on search requests.
- **Password administrator** is allowed to administer user passwords; for example, allowing a help desk to reset a user's password in the LDBM and TDBM backends.
- **Replication administrator** is allowed to administer advanced replication configurations.
- **Root administrator** is a super user (sum of all administrative roles, excluding no administrator role). This authority is equivalent to the authority of the **adminDN** in the LDAP server configuration file.
- **Schema administrator** is allowed to administer the schema.
- **Server configuration group member** is allowed to administer entries under and including the **cn=configuration** suffix in the CDBM backend.

See “Administrative roles” for more information.

Administrative roles

Whether administrative roles are defined for administrative group members in LDAP or in RACF, the administrative role categories and what they can do are the same. Administrative roles are gathered at authentication time and retained until the user reauthenticates. Therefore, any changes to the roles are not reflected until a new authentication or bind takes place. Group gathering is not performed for administrative group members. The following are the administrative role categories supported in the z/OS LDAP server:

- **Directory data administrator** Users assigned this role have unrestricted access to the entries in the LDBM and TDBM backends and the **cn=ibmpolicies** suffix in the CDBM backend. **aclEntry** values are ignored if the bound user has this administrative role. ACL filters are still applied, but cannot augment any permissions explicitly denied by this role definition. The directory data administrator has the following authority in the directory:

– add, modify, and delete access to master server DN and advanced replication supplier entries of a consumer server when also assigned the schema administrator and server configuration group member administrative roles.

Note: This does not apply to the users specified in the **masterServerDN** or **peerServerDN** configuration options.

– complete access to the **cn=replication, cn=configuration** entry in the CDBM backend

– modify access to the **ibm-slapdAdminPW** attribute in their own configuration entry in the CDBM backend

– complete access to entries in the TDBM and LDBM backends. However, for setting the password of any entry, the directory data administrator must abide by the normal password policy rules (if there are any). Users in this role cannot change the password of other administrative group members that exist in the directory.

– read, search, and compare access to all entries in and under the **cn=configuration** suffix in the CDBM backend

- **ibm-slapdAdminDN, ibm-slapdAdminPW** and **ibm-slapdAdminGroupEnabled** attributes under the **cn=configuration** entry
- **ibm-slapdAdminPW** attribute under other local administrative group member entries, and
- passwords of advanced replication supplier server credential entries (master server DNs).

– read access to the **cn=schema** entry

– cannot add or delete entries under the **cn=replication,cn=configuration** entry

– update access to changelog (GDBM) entries

• **No administrator** – Users assigned this role have no administrative privileges. By defining this role the LDAP root administrator revokes all the administrative privileges of the administrative group member. The no administrator has the following authority in the directory:

– modify access to the **ibm-slapdAdminPW** attribute in their own configuration entry in the CDBM backend

– read, search, and compare access to all entries in and under the **cn=configuration** suffix in the CDBM backend except for

- **ibm-slapdAdminDN, ibm-slapdAdminPW** and **ibm-slapdAdminGroupEnabled** attributes under the **cn=configuration** entry
- **ibm-slapdAdminPW** attribute under other local administrative group member entries, and
- passwords of advanced replication supplier server credential entries (master server DNs).

– read access to the **cn=schema** entry

– access to entries in the LDBM and TDBM backends through normal ACL evaluation. ACL filters are still applied, but cannot augment any permissions explicitly denied by this role definition. See Chapter 24, “Using access control,” on page 433 for more information.

– read access to changelog (GDBM) entries

• **Operational administrator** – Users assigned this role can perform persistent searches. **aclEntry** values are ignored when processing this role. ACL filters do not apply. The operational administrator has the following authority in the directory:

– ability to include the **PersistentSearch** control on search requests

- modify access to **ibm-slapdAdminPW** attribute in their own configuration entry in the CDBM backend
- read access to all entries in and under the **cn=configuration** suffix in the CDBM backend except for
  - **ibm-slapdAdminDN, ibm-slapdAdminPW** and **ibm-slapdAdminGroupEnabled** attributes under the **cn=configuration** entry
  - **ibm-slapdAdminPW** attribute under other local administrative group member entries, and
  - passwords of advanced replication supplier server credential entries (master server DNs)
- read access to the **cn=schema** entry
- read access to entries in the LDBM and TDBM backends and entries under and including the **cn=ibmpolicies** suffix in the CDBM backend
- read access to changelog (GDBM) entries

- **Password administrator** – Users assigned this role can unlock the accounts of other users or change passwords of users in the LDBM or TDBM backends. Password policy constraints set by the server do not apply to users assigned the password administrator role. This role has access to the LDBM and TDBM backends through normal ACL evaluation. ACL filters are still applied, but cannot augment any permissions explicitly denied by this role definition. See Chapter 24, "Using access control," on page 433 for more information. The password administrator has the following authority in the directory:
  - modify access to the **ibm-slapdAdminPW** attribute in their own configuration entry in the CDBM backend
  - read, search, and compare access to all the entries in and under **cn=configuration** except
    - **ibm-slapdAdminDN, ibm-slapdAdminPW** and **ibm-slapdAdminGroupEnabled** attributes under the **cn=configuration** entry
    - **ibm-slapdAdminPW** attribute under other local administrative group member entries, and
    - passwords of advanced replication supplier server credential entries (master server DNs)
  - read, search, and compare access to the **cn=schema** entry
  - ability to unlock user accounts or set the password without the following normal password policy rules for all the users in backends except global administrative group members. This means users with the password administrator role have:
    - read, write, and search access to the **userpassword** attribute and read, write, search, and compare access to the **pwdChangedTime** attribute
    - ability to add, delete, and modify the **ibm-pwdAccountLocked** attribute only if the new value of the attribute is false
    - ability to delete **pwdFailureTime, pwdAccountLockedTime, pwdExpirationWarned**, and **pwdGraceUseTime** attribute values for all users
  - read access to changelog (GDBM) entries
- **Replication administrator** – Users assigned the replication administrator role can update advanced replication topology entries. This role has access to entries in the LDBM and TDBM backends (including basic replication **replicaObject** entries) through normal ACL evaluation. ACL filters are still applied, but cannot augment any permissions explicitly denied by this role definition. See

Chapter 24, "Using access control," on page 433 for more information. The replication administrator has the following authority in the directory:

- no access to master server DN and advanced replication supplier entries in a consumer server
- read, write, search, and compare access to the **cn=replication, cn=configuration** entry in the CDBM backend
- modify access to the **ibm-slapdAdminPW** attribute in their own configuration entry in the CDBM backend
- read access to all the entries in and under **cn=configuration** except
  - **ibm-slapdAdminDN, ibm-slapdAdminPW** and **ibm-slapdAdminGroupEnabled** attributes under the **cn=configuration** entry
  - **ibm-slapdAdminPW** attribute under other local administrative group member entries, and
  - passwords of advanced replication supplier server credential entries (master server DNs).
- ability to issue the following advanced replication extended operations: **Cascading control replication, Control replication queue, Control replication, Control replication error log, Quiesce or unquiesce context**, and **Replication topology**
- read access to the **cn=schema** entry
- read access to changelog (GDBM) entries
- read, write, search, compare, add, and delete access to all advanced replication topology entries in the backends. The determination of replication entries is based on the presence of following objectclasses in the entry: **ibm-replicationcontext** (replication context), **ibm-replicagroup** (replica group), **ibm-replicasubentry** (replica subentry), **ibm-replicationAgreement** (replication agreement), **ibm-replicationCredentials** (credentials entry), **ibm-replicationCredentialsSimple** (credentials entry), **ibm-replicationCredentialsExternal** (credential entry), **ibm-replicationWeeklySchedule** (replication schedule), **ibm-replicationDailySchedule** (replication schedule), and **ibm-replicationFilter** (replication filter).

- **Root administrator** – Users assigned this role have the same access to data in the LDAP server as the LDAP root administrator (**adminDN** in the LDAP server configuration file). This role has permissions of all other roles except the no administrator role.
- **Schema administrator** – Users assigned this role have unrestricted access to the **cn=schema** entry only. This role has access to the LDBM and TDBM backends through normal ACL evaluation. ACL filters are still applied, but cannot augment any permissions explicitly denied by this role definition. See Chapter 24, "Using access control," on page 433 for more information. The schema administrator has the following authority in the directory:
  - modify access to the **cn=schema** entry
  - add, modify, and delete access to the master server DN and replication supplier entries (**cn=configuration** in the CDBM backend) of a consumer replica when the user also has the directory data administrator and server configuration group member roles
  - modify access to the **ibm-slapdAdminPW** attribute in their own configuration entry in the CDBM backend
  - read access to all the entries in and under **cn=configuration** except

- **ibm-slapdAdminDN, ibm-slapdAdminPW** and **ibm-slapdAdminGroupEnabled** attributes under the **cn=configuration** entry
- **ibm-slapdAdminPW** attribute under other local administrative group member entries, and
- passwords of advanced replication supplier server credential entries (master server DNs)

– read, write, search, and compare access to the **cn=schema** entry

– read access to changelog (GDBM) entries

- **Server configuration group member** – Users assigned this role have restricted update access to the **cn=configuration** entries in the CDBM backend. This role has access to entries in the LDBM and TDBM backends through normal ACL evaluation. ACL filters are still applied, but cannot augment any permissions explicitly denied by this role definition. See Chapter 24, "Using access control," on page 433 for more information. The server configuration group member has following authority in the directory:

  – add, modify, and delete access to all **cn=configuration** entries in the CDBM backend, except the following:
    - **ibm-slapdAdminDN** and **ibm-slapdAdminGroupEnabled** attributes under the **cn=configuration** entry
    - entries under the **cn=admingroup,cn=configuration** entry

  – add, modify, and delete access to the master server DN and replication supplier entries in the **cn=configuration** entry in the CDBM backend of a consumer replica when the user is also assigned directory data administrator and schema administrator roles

  – modify access to the **ibm-slapdAdminPW** attribute in its own configuration entry in the CDBM backend

  – read access to all the entries in and under **cn=configuration** except
    - **ibm-slapdAdminDN, ibm-slapdAdminPW**, and **ibm-slapdAdminGroupEnabled** attributes under the **cn=configuration** entry
    - **ibm-slapdAdminPW** attribute under other local administrative group member entries, and
    - passwords of advanced replication supplier server credential entries (master server DNs).

  – read access to the **cn=schema** entry

  – read access to changelog (GDBM) entries

Enabling the administrative group and roles

The server compatibility level must be 7 or greater and the CDBM backend must be configured to use the administrative group and roles. See "serverCompatLevel {3 | 4 | 5 | 6 | 7| 8}" on page 125 for more information about the **serverCompatLevel** configuration option. When the LDAP server compatibility level is 7 or greater, the following entries are automatically created in the CDBM backend, if they do not exist:

- **cn=admingroup,cn=configuration**
- **cn=safadmingroup,cn=configuration**

By default, the administrative group is not enabled in the LDAP server because the **ibm-slapdAdminGroupEnabled** attribute is set to **false** automatically in the **cn=configuration** entry, if the attribute does not exist. If the **ibm-slapdAdminGroupEnabled** attribute is set to **true**, group members can be added to member entries under the **cn=admingroup,cn=configuration** entry or added as

member attribute values to the **cn=safadmingroup,cn=configuration** entry. If the **ibm-slapdAdminGroupEnabled** attribute is deleted, the LDAP server treats the attribute as if it is set to **false**. The **member** attribute can be used to specify a RACF group as an administrative group member. This allows RACF administrators to assign administrative roles to all members of a RACF group.

The following **ldapsearch** utility command can be used to query the status of the **ibm-slapdAdminGroupEnabled** attribute value in the **cn=configuration** entry:

```
ldapsearch -D binddn -w passwd -s base -b cn=configuration "objectclass=*" ibm-slapdAdminGroupEnabled
```

The following **ldapmodify** utility command can be used to set the **ibm-slapdAdminGroupEnabled** attribute value to **true** in the **cn=configuration** entry. When successful, this activates the administrative group and roles feature:

```
ldapmodify -D binddn -w passwd -f file.ldif
```

where, the *file.ldif* contents are:

```
dn: cn=configuration
changetype: modify
replace: ibm-slapdAdminGroupEnabled
ibm-slapdAdminGroupEnabled: true
```

See "CDBM backend configuration and policy entries" on page 139 for more information about the above entries and attribute values that affect the administrative group and roles configuration.

Defining administrative group and roles

The administrative group member entry is defined in the LDAP server in the CDBM backend, however, the roles that are assigned to members of these groups can exist either in the LDAP server or in RACF. This section describes the necessary entries for defining the administrative group and roles in the LDAP server or in RACF.

Note: When the administrative group is enabled by using **ibm-slapdadmingroupenabled** that is set to **true**, there are special rules that protect the list of members that are defined in CDBM. See "Administrative roles" on page 159 for the rules. When the administrative group is not enabled, regular ACLs protect the list of members that are defined in CDBM.

Administrative roles defined in LDAP

An LDAP root administrator creates the administrative group member entry in LDAP for each administrative user under the **cn=admingroup,cn=configuration** entry in the CDBM backend. The administrative group member entry must have an object class value of **ibm-slapdAdminGroupMember** and contain the members and one or more administrative roles the user is allowed to have in the LDAP server. See Table 15 on page 165 for the optional and required attribute types.

Table 15. ibm-slapdAdminGroupMember objectclass schema definition (optional and required attributes)

Attribute description and example
ibm-slapdAdminDN A required attribute that specifies the bind distinguished name (DN) for this administrative group member. If this distinguished name points to an entry that exists in the LDBM, TDBM, or SDBM backend and a simple, CRAM-MD5, or DIGEST-MD5 bind is done, the password specified in that entry is checked during authentication. If this distinguished name does not exist in an entry in the LDBM, TDBM, or SDBM backend and a Kerberos or EXTERNAL bind is done, the alternate DN from a Kerberos or EXTERNAL bind or the DN extracted from the client's SSL certificate are checked during authentication. The **ibm-slapdAdminPW** attribute value must be specified if the distinguished name specified does not exist in the LDBM, TDBM, or SDBM backend and a simple authentication is being performed. Example: `ibm-slapdAdminDN: cn=dAdmin`
ibm-slapdAdminPW This optional attribute specifies the bind password for the distinguished name specified in the **ibm-slapdAdminDN** attribute value. If password policy is activated in the LDAP server, password policy rules do not apply to this password value. This password value is encrypted or hashed if it is added or modified when the **pwEncryption** configuration option is set to a value other than **none** in the CDBM backend. If **pwEncryption** is set to a value other than **none**, directory security improves because the password is no longer stored in the directory in clear text. This value should not be specified if the distinguished name specified in the **ibm-slapdAdminDN** attribute value exists in an LDBM, TDBM, or SDBM backend. Also, do not specify this value if the **ibm-slapdAdminDN** is an alternate DN from a Kerberos or SASL EXTERNAL bind or the DN from a client SSL certificate. **Note:** This value is only used if the entry specified in the **ibm-slapdAdminDN** attribute value exists under a configured suffix in the LDAP server. Example: `ibm-slapdAdminPW: secret`

ibm-slapdAdminRole
 	
A multi-valued attribute that specifies the administrative roles associated with the administrative group member. The valid values are:
• **DirDataAdmin** (for Directory administrator)
• **NoAdmin** (for No administrator)
• **OperationalAdmin** (for Operational administrator)
• **PasswordAdmin** (for Password administrator)
• **ReplicationAdmin** (for Replication administrator)
• **RootAdmin** (for Root administrator)
• **SchemaAdmin** (for Schema administrator)
• **ServerConfigGroupMember** (for Server configuration group member)
 	
If more than one role must be specified for this member, the **ibm-slapdAdminRole** attribute must be specified for each role. See "Administrative roles" on page 159 for more information about administrative roles.
 	
Example:
`ibm-slapdAdminRole: DirDataAdmin`
`ibm-slapdAdminRole: SchemaAdmin`

Note: The distinguished name of any created LDAP administrative group member entry cannot have the same value as the **adminDN, masterServerDN**, or **peerServerDN** configuration options or the distinguished names of any basic or advanced replication master server or advanced replication supplier server distinguished names. Also, these distinguished names cannot match any **member**

attribute values in the **cn=safadmingroup,cn=configuration** entry. A member that is an alternate DN is compared against other administrative group members.

Administrative roles defined in RACF

An LDAP root administrator adds the user to the **cn=safadmingroup,cn=configuration** entry as a **member** attribute value. The multi-valued **member** attribute must specify a distinguished name (DN) that eventually leads to a SAF user ID and is a member of the SAF administrative group. Examples of these distinguished names are: SDBM entries, DNs from an SSL client certificate, DN of a TDBM or LDBM entry participating in native authentication, and a Kerberos mapped DN. These users must then be given READ access to one or more administrative roles defined to RACF in LDAP general resource class. See Table 13 on page 144 for more information about the optional and required attribute types for the **cn=safadmingroup,cn=configuration** entry. The **member** attribute can also be used to specify a RACF group as an administrative group member. This allows RACF administrators to assign administrative roles to all members of a RACF group.

Note: The **member** attribute in the SAF administrative group member entry is not allowed to be set to values of the **adminDN, masterServerDN**, or **peerServerDN** configuration options or the distinguished names of any basic or advanced replication master server or supplier distinguished names. Also, the **member** attribute is not allowed to be set to any distinguished names that are created under the **cn=admingroup,cn=configuration** entry. A member that is an alternate DN is compared against other administrative group members.

When administrative group members are defined in the **cn=safadmingroup,cn=configuration** entry, the administrative roles associated with each member must also be defined in RACF profiles under the RACF supplied LDAP class. The LDAP class profiles must be defined as:

`Domain_Name`.ADMINROLE.`role` where:

- `Domain_Name` is the value specified in the **ibm-slapdSAFSecurityDomain** attribute of the **cn=configuration** entry. The LDAP server uses the value in this attribute as the first part of the RACF defined administrative role when it does SAF authorization checking on security-related profiles for the administrative roles. By default, the **ibm-slapdSAFSecurityDomain** attribute is set to **GLDSEC**. An LDAP root administrator is allowed to modify the **ibm-slapdSAFSecurityDomain** attribute to a new value. The value is converted to uppercase during SAF authorization checking. See Table 10 on page 139 for more information about the **ibm-slapdSAFSecurityDomain** attribute.
- ADMINROLE – This value must be the second value in the RACF profile name to indicate that it is an administrative role. A period must precede and follow this value.
- `role` - contains the assigned administrative role value. Values can be any of the following predefined values:
  - CONFIG (for Server configuration group member)
  - DIRDATA (for Directory data administrator)
  - NOADMIN (for No administrator)
  - OPER (for Operational administrator)
  - PASSWD (for Password administrator)
  - REPL (for Replication administrator)
  - ROOT (for Root administrator)
  - SCHEMA (for Schema administrator)

The SAF users that are to be granted administrative roles must be granted READ access to the LDAP class resource profile. See "Administrative group member examples" for an example of defining administrative roles in RACF.

Administrative group member examples

The following example adds the administrative group member under the **cn=admingroup,cn=configuration** entry and assigns password administrator and directory administrator roles to the member, `cn=dpAdmin`, that has a password value of `secret`. This example uses the **ldapadd** utility to add the **cn=admindp,cn=AdminGroup,cn=configuration** entry.

```
ldapadd -D "cn=admin" -w xxxx -f admingroup.ldif
```

where `admingroup.ldif` contains:

```
dn: cn=admindp,cn=AdminGroup,cn=configuration
changetype: add
objectclass: ibm-slapdAdminGroupMember
ibm-slapdAdminDN: cn=dpAdmin
ibm-slapdAdminPW: secret
ibm-slapdAdminrole: dirDataAdmin
ibm-slapdAdminrole: PasswordAdmin
```

The following example adds the administrative group member, `racfid=pAdmin,profiletype=user,cn=myracf`, to the **cn=safadmingroup,cn=configuration** entry and assigns the password administrator role to the member. The user ID, `pAdmin`, is an existing SAF user ID. Also, this illustrates an example when using native authentication and using RACF to assign roles. See "Example of setting up native authentication" on page 408 for more information. This example uses the **ldapmodify** utility to modify the **cn=safadmingroup,cn=configuration** entry.

```
ldapmodify -D "cn=admin" -w xxxx -f modSafAdminGroup.ldif
```

where `modSafAdminGroup.ldif` contains:

```
dn: cn=safadmingroup,cn=configuration
changetype: modify
add: member
member: racfid=pAdmin,profiletype=user,cn=myracf
member: cn=User1,ou=END,o=IBM,c=US
```

and the `cn=User1,ou=END,o=IBM,c=US` entry contains `ibm-nativeId` set to `pAdmin`.

If the **ibm-slapdSAFSecurityDomain** attribute in the **cn=configuration** entry is set to **GLDSEC**, the following profile in the LDAP class must be created in RACF for the password administrative role:

```
RDEFINE LDAP GLDSEC.ADMINROLE.PASSWD UACC(NONE)
```

The `pAdmin` ID must be granted READ access to this profile to have the password administrator role:

```
PERMIT GLDSEC.ADMINROLE.PASSWD CLASS(LDAP) ID(pAdmin) ACCESS(READ)
```

The following example adds all members of a RACF group, **pAdGrp**, to the administrative group. The DN for that group is: `racfid=pAdGrp,profiletype=group,cn=myracf`. It must be added to the **cn=safadmingroup,cn=configuration** entry. The group ID, `pAdGrp`, is an existing RACF group ID. This example uses the **ldapmodify** utility to modify the **cn=safadmingroup,cn=configuration** entry.

```
ldapmodify -D "cn=admin" -w xxxx -f modSafAdminGroup2.ldif
```

where `modSafAdminGroup2.ldif` contains:

```
dn: cn=safadmingroup,cn=configuration
changetype: modify
add: member
member: racfid=pAdGrp,profiletype=group,cn=myracf
```

The `pAdGrp` ID must be granted READ access to this profile to have the password administrator role:

```
PERMIT GLDSEC.ADMINROLE.PASSWD CLASS(LDAP) ID(pAdGrp) ACCESS(READ)
```

Note: If the **dsconfig** utility was used to configure the z/OS TDS server and the RACF job was run, the following commands were run:

```
SETROPTS GENERIC(LDAP)
RDEFINE LDAP GLDSEC.ADMINROLE.* UACC(NONE)
RDEFINE LDAP GLDSEC.ADMINROLE.NOADMIN UACC(NONE)
SETROPTS CLASSACT(LDAP)
```

If you activate the administrative group and roles and define a user as a member in **cn=safadmingroup,cn=configuration**, a RACF message is displayed on the z/OS console and job log for any role that the member does not have READ access to.

Administrative roles and extended operations

Table 16 lists the LDAP server's supported extended operations and the administrative roles. The intersection specifies whether a specific administrative role has authority for the extended operation. An LDAP root administrator and master server DN (advanced replication) have authority to all extended operations. See Appendix C, "Supported server controls," on page 679 and Appendix D, "Supported extended operations," on page 695 for more information about the supported server controls and extended operations.

Table 16. Administrative roles authorized to issue various extended operations and controls

Extended operation or control	Directory admin	Replication admin	Schema admin	Server configuration group member	Password admin	No admin	Operational admin	Root admin
Account status extended operation	Yes	No	No	No	No	No	No	Yes
Cascading control replication extended operation	Yes	Yes	No	No	No	No	No	Yes
changeLogAddEntry extended operation	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Control replication extended operation	Yes	Yes	No	No	No	No	No	Yes
Control replication error log extended operation	Yes	Yes	No	No	No	No	No	Yes
Control replication queue extended operation	Yes	Yes	No	No	No	No	No	Yes
Effective password policy extended operation	Yes	No	No	No	Yes	No	No	Yes
GetDnforUserid extended operation	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
GetEffectiveAcl extended operation	Yes	No	Yes	Yes	No	No	No	Yes
GetPrivileges extended operation	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
PersistentSearch control	No	No	No	No	No	No	Yes	Yes
Quiesce or unquiesce context extended operation	Yes	Yes	No	No	No	No	No	Yes
Remote auditing extended operation	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*
Remote authorization extended operation	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*
RemoteCryptoCCA extended operation	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*
RemoteCryptoPKCS#11 extended operation	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*
Replication topology extended operation	Yes	Yes	No	No	No	No	No	Yes

Table 16. Administrative roles authorized to issue various extended operations and controls (continued)

Extended operation or control	Directory admin	Replication admin	Schema admin	Server configuration group member	Password admin	No admin	Operational admin	Root admin
Start TLS extended operation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
unloadRequest extended operation	Yes	No	Yes, only if unloading schema	No	No	No	No	Yes
User type extended operation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Note: *The administrative roles have no effect on the authority to run the **Remote auditing, Remote authorization**, **RemoteCryptoCCA**, or **RemoteCryptoPKCS#11** extended operations. The bound user must map to a RACF user that has the appropriate authority. See ICTX plug-in and Remote crypto plug-in for more information.

Administrative group and roles-related extended operation

The **User type** extended operation is provided to allow a user to determine their user type and user roles. The **ldapexop** utility enables the call to the **User type** extended operation. Table 17 summarizes the extended operation. See "ldapexop utility" on page 255 for more information.

Table 17. Administrative group and roles extended operation

ldapexop operation	ldapexop description	Overview
getusertype	**User type** extended operation	This extended operation is used by a user to determine their user type and user roles.

User type extended operation examples

In this example, an LDAP root administrator, **cn=admin**, issues the **User type** extended operation:

```
ldapexop -D cn=admin -w secret -op getusertype
```

```
User    : root_administrator
Role(s) : operational_administrator schema_administrator replication_administrator
 password_administrator directory_data_administrator server_config_administrator
```

In this example, `cn=barb,o=acme,c=us` issues the **User type** extended operation, is a member of the administrative group, and is assigned the no administrative role.

```
ldapexop -D cn=barb,o=acme,c=us -w barbpw -op getusertype
```

```
User : admin_group_member
Role(s) : no_administrator
```

In this example, an unauthenticated user issues the **User type** extended operation:

```
ldapexop -op getusertype
```

```
User    : ldap_user_type
Role(s) : ldap_user_role
```

Chapter 10. Running the LDAP server

This topic describes what is necessary to get the LDAP server running.

Setting up the PDSE 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 PDSE format only. In order for these DLLs to be located by the LDAP server at run time, the PDSE which contains these DLLs (SYS1.SIEALNKE) must be in the LNKLST (the default installation) or referenced in a **STEPLIB DD** card. In addition, the SYS1.SIEALNKE data set must be APF-authorized. If a **STEPLIB DD** statement is used, all data sets in the concatenation must be APF-authorized. The LDAP server also requires the *hlq*.SCEERUN and *hlq*.SCEERUN2 data sets. Both of these data sets must be APF-authorized.

Setting up and running the LDAP server

The LDAP server must be run as a started task. To do this, you must define the started task for the LDAP server and then you can run the LDAP server by using JCL. The LDAP server can be run in 31-bit mode or 64-bit mode.

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 47), you must define the DSSRV started task. The examples and the sample startup procedure use the name DSSRV for this task, but you can use any name for it.

To define the started task for the user ID you created, you can use the following RACF commands.

```
RDEFINE STARTED DSSRV.** STDATA(USER(LDAPSRV))
SETROPTS RACLIST(STARTED) REFRESH
```

Note: When using **dsconfig** to configure the LDAP server, the started task is already defined.

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 the DSSRV member of *GLDHLQ*.SGLDSAMP on the system where the LDAP server is installed. If you have a ServerPac installation, *GLDHLQ* is **GLD**. Use DSSRV for starting the LDAP server in 31-bit or 64-bit mode. The JCL procedure can be started in the System Display and Search Facility (SDSF) or from the operator's console after the sample JCL is placed into the installation-specific library for procedures.

The JCL must be tailored before it can be run. In particular, you must change the program name to GLDSRV31 to run the LDAP server in 31-bit mode or GLDSRV64 to run in 64-bit mode. When you are using 31-bit mode, make sure the REGION= parameter is the appropriate size. When you are using 64-bit mode with one or more LDBMs, make sure the MEMLIMIT= parameter is coded with the appropriate size. If it is too small, then the server might not start and the following messages are issued:

```
GLD1106E LDBM backend initialization failed for backend named LDBM-0001.
GLD1101A Unable to load the database backends.
GLD1007I LDAP server is stopping.
```

See “LDBM performance considerations” on page 606 for size estimates.

For 64-bit mode, the MEMLIMIT= parameter should be coded such that if an LDBM (or more) is included, then the MEMLIMIT size needs to be considered appropriately.

To start the LDAP server in SDSF, enter:

```
/s dssrv
```

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

```
s dssrv
```

The LDAP server has the following optional command-line parameters. One or more of these might be specified when starting the LDAP server.

-f *pathname*
: Name of LDAP server configuration file to be read. The file name is case-sensitive unless it refers to a data set or DD statement. A data set is indicated by `//'dataset-name'` while a DD statement is indicated by `//DD:dd-name`. The configuration data set specified by the **CONFIG DD** statement is used if the **-f** parameter is not specified. The **/etc/ldap/ds.conf** configuration file is used if the **-f** parameter is not specified and the **CONFIG DD** statement is not defined.

-l *ldap_URL*
: Host name or IP address and port number on which the LDAP server binds and listen for incoming requests. See the **listen** parameter at “listen ldap_URL” on page 104 for information about the *ldap_URL* parameter. The **-l** parameter can be specified multiple times to add more *ldap_URL* values. The values that are specified using the **-l** command line parameter override the values that are specified for the **listen** option in the LDAP server configuration file.

  You should reserve the port number or numbers that are chosen here in your TCP/IP profile data set. Also, be aware that port numbers below 1024 might require additional specification. See *z/OS V2R2.0 Communications Server: IP Configuration Guide* for more information.

-m
: Start the server in maintenance mode rather than normal mode. Maintenance mode severely restricts access to the server. See “Basic replication maintenance mode” on page 475 or “Advanced replication maintenance mode” on page 536 for more information.

-p *port*

  Note: The **port** option is deprecated when the **listen** option is specified. See “listen ldap_URL” on page 104 for information about the **listen** option. Port number where the LDAP server listens for nonsecure communications. If you specify this parameter, it overrides the **port** value that might be in the LDAP server configuration file if there are not any **listen** values specified in the configuration file or **-l** parameters on the command line.

-s *secureport*

Note: The **secureport** option is deprecated when the **listen** option is specified. See “listen ldap_URL” on page 104 for information about the **listen** option.
Port number where the LDAP server listens for secure communications. If you specify this parameter, it overrides the **securePort** value that might be in the LDAP server configuration file if there are not any **listen** values specified in the configuration file or **-l** parameters on the command line.

-d *debug_level*

The debug level is a mask that might 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 ’x8000+x8’, or ’ERROR+CONNS’
  - '2147483647-16' is the same as specifying 'x7FFFFFFF-x10' or 'ANY-BER'
  - By beginning the debug level with a minus sign, you can deactivate debug collection for a debug level. For example, "-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 a debug level. For example, "+CONNS" modifies an existing debug level by activating connection traces.

Note: Specifying the debug level by using decimal or hex values with a plus or minus sign is not necessarily the same as specifying the sum or difference as the debug level. For example, specifying '7+1' activates the 'TRACE', 'PACKETS', and 'ARGS' debug levels, while specifying '8' activates only the 'CONNS' debug level. Similarly, specifying '16-1' activates only the 'BER' debug level, while specifying '15' activates 'TRACE', 'PACKETS', ARGS', and 'CONNS'.

Table 18 lists the debug levels and related decimal, hexadecimal, and keyword values. Keywords can be abbreviated using the uppercase characters for each keyword.

Table 18. LDAP 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
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

Table 18. LDAP debug levels (continued)

Keyword	Decimal	Hexadecimal	Description
REPLication	1024	0x00000400	Replication activities
PARSe	2048	0x00000800	Parsing activities
PERFormance	4096	0x00001000	Performance statistics
SDBM	8192	0x00002000	Backend activities (SDBM)
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
BECApabilities	4194304	0x00400000	Backend capabilities
CACHe	8388608	0x00800000	Cache activities
INFO	16777216	0x01000000	General processing information
LDBM	33554432	0x02000000	Backend activities (LDBM)
PLUGin	67108864	0x04000000	Plug-in extension 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.
- Before you start the server, the **LDAP_DEBUG** environment variable might be set. The server uses this value first. For example,

  ```
  export LDAP_DEBUG='ERROR+TRACE'
  ```

- When you start the server, the **-d** parameter might be specified. The debug level that is specified on this parameter replaces, adds to, or deletes from the preceding debug level. For example,
  - `s dssrv,parms='-d ERROR'`

    replaces the current debug level that is off or has been set by the **LDAP_DEBUG** environment variable with the new debug level of only ERROR.
  - `s dssrv,parms='-d +ERROR'`

    adds the ERROR debug level to the current debug level that is off or has been set by the **LDAP_DEBUG** environment variable.
  - `s dssrv,parms='-d -ERROR'`

    removes the ERROR debug level from the current debug level that is off or has been set by the **LDAP_DEBUG** environment variable.
- It is possible to change the debug level when the server is running. See "Dynamic debugging" on page 182 for more information.

-t *transition mode*

Starts the server in transition mode. Transition mode is used to update the server compatibility level, or to add or remove shared backends in a sysplex configuration without incurring a complete LDAP service outage. The server running in transition mode provides uninterrupted LDAP services while other servers are being updated and are unavailable.

Notes:

1. Only one server can be started and running with transition mode in the whole sysplex.
2. Transition mode is a temporary work mode used during sysplex level configuration updates. Transition mode is removed and changed to normal work mode automatically when this server becomes sysplex owner and the configuration updates complete.
3. Transition mode is supported for z/OS Version 2 Release 2 and above. See "Updating LDAP configurations settings in a sysplex without server outage" on page 210 for more information.

When the LDAP server has been started and is ready, GLD1004I is displayed:

```
GLD1004I LDAP server is ready for requests.
```

"Running the LDAP server using data sets" on page 176 contains information about using a data set for the LDAP server configuration file. To specify the configuration file as a data set name or a DD name in SDSF, some special syntax is necessary.

To specify a full data set name, it might 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.DS.CONF`, the start command for the LDAP server in expanded input SDSF or the console is:

```
s dssrv,parms='-f //''''MYUSERID.DS.CONF'''''
```

or, if more parameters are needed:

```
s dssrv,parms='-f //''''MYUSERID.DS.CONF'''' -l ldap://:999'
```

If a **DD** name, **CONFIG**, was established in the DSSRV procedure, as follows:

```
CONFIG DD DSN=MYUSERID.DS.CONF,DISP=SHR
```

the LDAP server can be started from expanded input SDSF or the console by entering:

```
s dssrv,parms='-f //DD:CONFIG'
```

To stop the LDAP server in SDSF, enter:

```
/p dssrv
```

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

```
p dssrv
```

This command causes the LDAP server to shut down.

LDAP server messages and debug output

The LDAP server writes messages to **stdout** and **stderr**. Messages sent to **stdout** and **stderr** appear in **DD:DSOUT** in the provided JCL when running the LDAP server. **DSOUT** 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 about how to use SDSF.

Output from the LDAP server debug facility is directed to the file specified by the **LDAP_DEBUG_FILENAME** environment variable. If this environment variable is not set, the output is sent to **stdout**, which is redirected to **DSOUT** as explained above.

Running the LDAP server using data sets

Note: Using the LDAP configuration utility (**dsconfig**) to configure your server creates all the necessary files in a partitioned data set.

The LDAP server accepts several of its files as data sets. Data set versions of the configuration file and the environment variables file are not shipped with the LDAP server, but can be created by using the **OGET** command to copy the file system versions of the files into data sets. (See *z/OS UNIX System Services Command Reference* for information about the use of the **OGET** command.)

The default data set characteristics for record format and record length (V 255) which **OGET** uses 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 before performing the **OGET** operation.

A data set version of the DSNAOINI file needed for the TDBM and GDBM (when DB2-based) backends can be created by copying and editing the default file provided by DB2. See step 4 on page 21. The **DSNAOINI** file name can be specified in the **dsnaoini** option in the LDAP server configuration file, in a **DSNAOINI DD** statement in the DSSRV procedure, or in a **DSNAOINI** environment variable. The **DD** statement takes precedence, followed by the environment variable, and then the configuration option.

Note: Be sure that use of sequence numbers is turned off when editing this data set.

When the data set versions of these files are available, they can be specified in the DSSRV procedure. The configuration file can be specified by using the **CONFIG DD** statement, the environment variables file can be specified by using the **ENVVAR DD** statement, and the **DSNAOINI** file can be specified by using the **DSNAOINI DD** statement.

Verifying the LDAP server

The following examples show how you can verify your LDAP server by using the **ldapsearch** utility. Note you can use any LDAP client to do this.

- **Verifying TDBM and LDBM**

  In the command below, substitute the **suffix** value from your LDAP server 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:

1. If **allowAnonymousBinds off** is specified in the LDAP server configuration file, you must specify a distinguished name to bind with using the **-D** and **-w** options on the **ldapsearch** utility.
2. The LDAP search returns the message "No such object" if the suffix entries have not been loaded into the directory. The TDBM or LDBM suffix entries can be added by using the steps outlined in "Finalizing setup of LDAP backends" and then this LDAP search can be tried again to verify that the entry is correctly added.

- **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, replace `cn=myRacf` with 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 root administrator DN or another DN authorized to search the change log.

  ```
  ldapsearch -h 127.0.0.1 -D binddn -w passwd -s base -b cn=changelog "objectclass=*"
  ```

- **Verifying CDBM**

  For CDBM, you must bind with the LDAP root administrator DN or another DN authorized to search the `cn=ibmpolicies` and `cn=configuration` CDBM suffixes.

  ```
  ldapsearch -h 127.0.0.1 -D binddn -w passwd -s base -b cn=ibmpolicies "objectclass=*"
  ldapsearch -h 127.0.0.1 -D binddn -w passwd -s base -b cn=configuration "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 this can be done from TSO also by substituting **LDAPSRCH** for **ldapsearch**.

See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information about **ldapsearch**.

Finalizing setup of LDAP backends

To finalize setup of LDAP backends, follow these steps:

1. If you have configured an LDBM or TDBM backend, modify the LDAP server schema entry to contain the schema needed for your usage of these backends. The **schema.user.ldif** and **schema.IBM.ldif** files found in the **/usr/lpp/ldap/etc** directory might contain the schema you need. **schema.IBM.ldif** requires that you first load **schema.user.ldif**. Additional schema might also be needed by the applications that are going to use the LDBM or TDBM directory. See Chapter 15, "LDAP directory schema," on page 275 for more information.

   Note:

a. The distinguished name (DN) of the LDAP server schema is `cn=schema`. If the ldif file containing your schema has a DN of `cn=schema,`*`suffix`*, then update the file to change the DN to `cn=schema`.
b. Do not add any additional schema to use the GDBM, CDBM, or SDBM backends unless you are using SDBM to access RACF customized fields or adding user-defined entries to CDBM. The initial schema built into the LDAP server is typically sufficient for these backends.

2. Load the suffix entries for each configured LDBM and TDBM backend. Do not try to load a suffix entry for the GDBM, CDBM, or SDBM backend. The GDBM, CDBM, and SDBM suffix entries are generated automatically by the LDAP server.

   Note: If you intend to load large amounts of data in LDIF format into a TDBM directory, see "ldif2ds utility" on page 235 for instructions on using the **ldif2ds** 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 **ldif2ds**.
   Use the **ldapadd** utility to load the suffix entry.

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

   where *file* is a file containing the entries to be loaded in LDIF format. For example, if `suffix.ldif` contains the following suffix entry:

   ```
   dn: o=Your Company
   objectclass: organization
   o: Your Company
   ```

   then the suffix entry can be added by **ldapadd** utility as follows:

   ```
   ldapadd -h myhost -p 389 -D cn=admin -w secret -f suffix.ldif
   ```

   See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information about the **ldapadd** utility.

3. Load additional entries into the LDBM, TDBM, or CDBM backend, as you want. You can use the **ldapadd** utility to load entries into these backends. For loading many entries into TDBM, consider using the **ldif2ds** utility. See "ldif2ds utility" on page 235 for more information. You must load entries in hierarchical order and an entry cannot be loaded before its parent entry is loaded.

   You can also add entries to SDBM (using **ldapadd**), but this results in adding profiles to RACF. See Chapter 17, "Accessing RACF information," on page 327 for more information. You cannot load entries into the GDBM backend. GDBM entries are only created by the LDAP server itself.

4. If GDBM is configured, set an appropriate ACL for controlling access to change log entries in the GDBM backend. See Chapter 28, "Change logging," on page 563 for more information.

5. If CDBM is configured, set an appropriate ACL for controlling access to the **cn=ibmpolicies** suffix because, by default, all users have access to that suffix. The ACL set on the **cn=configuration** suffix only allows access to the LDAP root administrator or an administrator with the appropriate authority. See "Administrative roles" on page 159 for more information about administrative role authority. Also, see Chapter 26, "Advanced replication," on page 487 and Chapter 18, "Password policy," on page 365 for more information.

6. After initial set-up of the LDAP server, you might want to remove the **adminPW** configuration option from the LDAP server configuration file. See "Establishing the root administrator DN and basic replication replica server DN and passwords" on page 151 for more information.

Environment variables used by the LDAP server

There are a number of environment variables that are processed by the LDAP server and utilities. Except for **LDAP_DS_ENVVARS_FILE**, they can be specified in the LDAP server environment variables file. By default, the file name is **/etc/ldap/ds.envvars**. The name can be reset by using the **LDAP_DS_ENVVARS_FILE** environment variable or, if that is not set, by the **ENVVAR DD** in the procedure that is used to start the LDAP server. Environment variables are read once, during LDAP server initialization. The LDAP server must be stopped and restarted to put a change to an environment variable into effect.

Below are some rules for setting up the environment variables file:

- The file must be in code page IBM-1047.
- An environment variable line consists of *name=value*, starting in column one. Blanks at the end of each line are removed, but no other blanks are removed.
- A line can be continued by putting a backslash (\) as the last non-blank character on the line. The backslash is removed and the next line is appended to the previous line. The maximum length of the initial line plus all continuation lines is 1024 characters.
- The *value* can be entirely enclosed in quotation marks (') or double quotation marks (") to include trailing blanks and to process a trailing backslash as part of the value. The quotation marks are removed from the value.
- A line that begins with a # in column one is a comment line and is ignored. A trailing backslash (\) on a comment line is ignored and the comment line is not continued.
- If the *name* corresponds to an environment variable that is already set and *value* is specified, the new value is ignored and the existing value is not changed. If *value* is not specified, the environment variable is deleted.
- Processing continues with the next line if any error occurs.

The list below describes the LDAP server environmental variables:

GLDLOG_MICROSECONDS=ON | *anything_else*
: Controls whether all generated activity log records contain microseconds in their time stamps. Microseconds are added to the time stamp if the value is set to **ON**. Microseconds are not included if the value is not **ON** or if the environment variable is not specified. The **GLDLOG_MICROSECONDS** environmental variable is deprecated. See "Activity logging" on page 193 or the **logFileMicroseconds** configuration option at "Configuration file options" on page 90 for more information.

GLDLOG_MSG=MSGS | NOMSGS
: Controls whether activity log records are generated when messages are created by the LDAP server. Messages are not written to the log if the environment variable is not specified. The **GLDLOG_MSG** environmental variable is deprecated. See "Activity logging" on page 193 or the **logFileMsgs** configuration option at "Configuration file options" on page 90 for more information.

GLDLOG_OPS=WRITEOPS | ALLOPS | SUMMARY
: Controls which operations generate LDAP server activity log records. No operations are logged if the environment variable is not specified. The **GLDLOG_OPS** environmental variable is deprecated. See "Activity logging" on page 193 or the **logFileOps** configuration option at "Configuration file options" on page 90 for more information.

GLDLOG_TIME=TIME | NOTIME | MERGEDRECORD
Controls whether LDAP server activity log records are generated when the operation being logged ends. Log records are not generated when an operation ends if this environment variable is not specified or is set to **NOTIME**. The **GLDLOG_TIME** environmental variable is deprecated. See "Activity logging" on page 193 or the **logFileRecordType** configuration option at "Configuration file options" on page 90 for more information.

IBMSLAPD_REPL_UPDATE_EXTRA_SECS=*interval*
Specifies, in seconds, how long the advanced replication engine waits for a replication operation to complete before setting an **LDAP_TIMEOUT** error code. By default, the advanced replication engine waits 60 seconds.

LDAP_ADVREPL_CLEANUP_INTERVAL=*interval*
Specifies, in seconds, how often backends participating in advanced replication delete replicated updates. If an incorrect value or 0 is specified, the value is set to 900 (delete replicated updates every 15 minutes). This is also the value that is used if the environment variable is not specified.

LDAP_COMPAT_FLAGS=*level*
Specifies the needed compatibility level setting. The value for **LDAP_COMPAT_FLAGS** is a mask that you can specify in the following ways:

- A decimal value (for example, `1`)
- A hexadecimal value (for example, `x02` or `X02`)
- A keyword (for example, `SDBM_MIXEDDN`)
- A construct of these values using plus and minus signs to indicate inclusion or exclusion of a value.

See "LDAP_COMPAT_FLAGS environment variable" on page 208 for more information.

LDAP_CONSOLE_LEVEL=I | W | E | A
Specifies the message severity level for sending a message that is created by the LDAP server to the operator console. Messages with a severity equal to or higher than the specified severity are sent to the operator console in addition to the normal output destination. Note that some LDAP server messages are always written to the operator console and are not affected by this value. Messages with a severity of E or higher are sent to the operator console if the environment variable is not specified.

LDAP_CTRACE_BUFFSIZE=*size*
Sets the amount of storage that allocates for CTRACE records within the LDAP server. The minimum value that can be specified is 1024000 bytes. This is also the value that is used if an incorrect value is specified or if the environment variable is not specified. On the average, each CTRACE record is about 120 bytes long. See "CTRACE in-memory trace records" on page 183 for more information.

LDAP_DEBUG=*level*
Specifies the needed debug level. The value for **LDAP_DEBUG** is a mask that you can specify in the following ways:

- A decimal value (for example, `32`)
- A hexadecimal value (for example, `x20` or `X20`)
- A keyword (for example, `FILTER`)
- A construct of these values using plus and minus signs to indicate inclusion or exclusion of a value.

See Table 18 on page 173 for more information.

LDAP_DEBUG_FILENAME=*filename* | **CTRACE_ONLY**
Specifies the fully qualified name of the LDAP debug output file. The debug output is written to **stdout** if this environment variable is not specified. The debug file is not used if LDAP debugging is not active. If using an output file, make sure that the file is not being used for any other purpose.

The current process identifier is included as part of the debug file name when the name contains a percent sign (%).

Example: If **LDAP_DEBUG_FILENAME** is set to /tmp/ldap.%.trc and the current process identifier is 247, the debug file name is /tmp/ldap.247.trc.

If the value is **CTRACE_ONLY**, then all debug output is only written to the internal CTRACE buffers.

LDAP_DS_ENVVARS_FILE=*filename*
Specifies the name of the LDAP server environment variables file. The file name in the **ENVVAR DD** in the start procedure for the LDAP server is used if the environment variable is not specified. If the **ENVVAR DD** is not specified, the file name defaults to **/etc/ldap/ds.envvars**.

LDAP_ERROR_LOGGING=STDOUT | STDERR | BOTH
Specifies how error messages are logged. The following values can be specified:

STDOUT Error messages are written to standard output as specified by the **LDAP_STDOUT_FILENAME** environment variable.

STDERR Error messages are written to standard error as specified by the **LDAP_STDERR_FILENAME** environment variable.

BOTH Error messages are written to both standard output and to standard error.

Error messages are written to standard error if this environment variable is not specified.

LDAP_NETWORK_POLL=*interval*
Specifies, in minutes, how often the LDAP server polls a network interface to determine whether it failed or becomes active. If 0 is specified, the value is set to 5 (poll every 300 seconds). This is also the value that is used if the environment variable is not specified.

LDAP_PRINT_CONFIG=1 | *anything_else*
Controls whether the configuration options used by the LDAP server are displayed when the LDAP server is started. The options are displayed if the value is 1 and are not displayed for any other value. The options are displayed if the environment variable is not specified.

LDBM_SHUTDOWN_FAST=1 | *anything_else*
Controls how a file-based backend (LDBM, CDBM, and file-based GDBM) in the LDAP server stops. Server shutdown typically frees all storage that is allocated and held by each backend. This can potentially be a large amount of storage for LDBM because it holds all its entries in memory, therefore, can be very time consuming. When the value is set to 1, storage that is allocated by the LDBM backend is not released before the LDAP server stops (the storage is eventually released by the operating system). When the value is not 1, the storage is freed before the LDAP server stops. This is also the processing that occurs if the environment variable is not specified.

LDAP_STDERR_FILENAME=*filename*
Specifies the fully qualified name of the file to receive standard error messages generated using LDAP message services. Messages are written to **stderr** if this environment variable is not specified. Make sure that the output file is not being used for any other purpose.

LDAP_STDOUT_FILENAME=*filename*
Specifies the fully qualified name of the file to receive standard output messages generated using LDAP message services. Messages are written to **stdout** if this environment variable is not specified. Make sure that the output file is not being used for any other purpose.

LDAP_TDBM_CACHEDELAY=*interval*
Specifies the number of seconds the LDAP server delays before it examines a TDBM DB2 database to detect out-of-date caches, rather than checking on every LDAP request. This can be used to reduce the cost of checking for up-to-date cache information. The caches affected are the referral cache, ACL (access control list) caches, and group cache. This environment variable is only used for a TDBM backend that is running in multiserver mode with a DB2 database that can be shared with a z/OS Integrated Security Services LDAP server (therefore, the DB_VERSION value of the database is less than '4.0' or the **serverCompatLevel** configuration option is **3**).

The valid range for the value is 0 to 2147483647. A value of 0 (zero) causes the DB2 database to be examined once per request. This is the default behavior if this environment variable is not specified.

This environment variable should not be set if there are applications that cannot tolerate temporarily out-of-date cache information in the LDAP server. It is best suited when most LDAP operations are search requests and the directory information is mostly static. It can also be useful even if there are many LDAP update operations, if the updates do not affect referrals, ACLs, or nested and dynamic groups.

LDAP_USE_INTERNAL_RNG=0 | *anything_else*
Specifies which random byte generator the LDAP server uses for the creation of the salt value for Salted SHA (SSHA) and Salted SHA-2 hashing and the generation of random data bytes for CRAM-MD5 and DIGEST-MD5 authentication binds. The LDAP server uses the SSL random byte generator that is provided in the **gsk_generate_random_bytes()** routine if the value is 0 or the environment variable is not specified. When the value is not 0, the LDAP server uses an internal random byte generator.

Dynamic debugging

When the LDAP server is running 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 if the command is entered from SDSF, it must be preceded by a slash (/). In the command:

```
f dssrv,debug debug_level
```

the *debug_level* is a mask that specifies the needed debug level. See Table 18 on page 173 for an explanation of the debug level values.

Debug information is 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*.

CTRACE in-memory trace records

CTRACE provides an in-memory tracing interface that is common with other z/OS products. The LDAP CTRACE support captures the following LDAP server output:

- All messages
- All debug message output for the ERROR debug level, regardless of debug level setting
- All debug message output for active debug levels. See Table 18 on page 173 for more information about debug levels

By always capturing ERROR debug output, the CTRACE support provides FFDC (First Failure Data Capture) to assist in problem debugging without the performance overhead of running with the debug facility enabled.

The default size of the storage used for the in-memory trace table is 1024000 bytes, which holds about 8000 CTRACE entries. The table wraps when the end is encountered, overwriting the oldest CTRACE entries. The trace table size can be increased by setting the **LDAP_CTRACE_BUFFSIZE** environment variable before the LDAP server is started. See "Environment variables used by the LDAP server" on page 179 for more information. There is no maximum trace table size except that imposed by the availability of system storage.

The LDAP CTRACE support is initialized during LDAP server startup. If CTRACE initialization fails, an error message is issued and server startup continues without CTRACE support. The LDAP CTRACE support cannot be turned off.

When the debug facility is active, debug output by default goes to both the CTRACE in-memory table and the output file indicated by the **LDAP_DEBUG_FILENAME** environment variable (or **stdout** if the environment variable is not set). If **LDAP_DEBUG_FILENAME=CTRACE_ONLY** is set, the debug output is only sent to CTRACE. The LDAP server **DEBUG** operator modify command can be used when the LDAP server is running to change whether debug output is sent only to CTRACE or to both CTRACE and the debug output file.

To direct debug output to CTRACE only, issue:

```
f dssrv,debug output=memory
```

To direct debug output to both CTRACE and to the debug output file, issue:

```
f dssrv,debug output=both
```

Viewing LDAP server CTRACE output

The LDAP server CTRACE component name is GLDSRVR. The LDAP server creates a subnode under GLDSRVR, identified by the hex address space identifier (ASID) of the LDAP server. Each LDAP server running on the system has its own subnode. The subnode is deleted when the LDAP server terminates as it typically does. To see the current subnodes, issue the following command from SDSF or the operator console:

```
d trace,comp=gldsrvr,sublevel,n=20
```

The results are like:

```
D TRACE,COMP=GLDSRVR,SUBLEVEL,N=20
IEE843I 13.49.52  TRACE DISPLAY
        SYSTEM STATUS INFORMATION
 ST=(ON,0256K,00512K) AS=ON  BR=OFF EX=ON  MT=(ON,064K)
  TRACENAME
  =========
  GLDSRVR
                       MODE BUFFER HEAD SUBS
                       ====================
                       OFF         HEAD    1
     NO HEAD OPTIONS
  SUBTRACE             MODE BUFFER HEAD SUBS
 ------------------------------------------------------
  ASID(004E)           OFF
     ASIDS      *NOT SUPPORTED*
     JOBNAMES   *NOT SUPPORTED*
     OPTIONS    *NONE*
     WRITER     *NOT SUPPORTED*
```

To view CTRACE records, an SDUMP, SADMP, or SYSMDUMP dump must be performed for the LDAP server address space. The CTRACE records in the resulting dump can be processed by the IPCS CTRACE facility, using the LDAP format table, GLDSCTFT, that resides in SYS1.SIEAMIGE. See *z/OS MVS IPCS Commands* for more information about IPCS. Here is an excerpt of the LDAP server CTRACE output (the message lines are wrapped here to fit on the page):

```
ctrace comp(gldsrvr) sub((asid(004e))) full

   COMPONENT TRACE FULL FORMAT
   COMP(GLDSRVR) SUBNAME((ASID(004E)))
   **** 05/20/2005

    SYSNAME   MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
    -------   --------  --------  ---------------  -------------

    DCESET3   ERROR     00000001  18:49:47.517060  LDAP_TRACE_ERROR

      060615 14:49:47.516866 GLD6012E Unable to open database file /usr/include/LDBM-1.db:
                                      141/05620060 - EDC5141I Read-only file system..
   ------------------------------------------------------------------------------
    DCESET3   ERROR     00000001  18:49:47.520275  LDAP_TRACE_ERROR

      060615 14:49:47.520124 GLD1106E GDBM backend initialization failed..
```

Displaying performance information and server settings

The LDAP server provides a **DISPLAY** operator modify command that is used to display information about the server, including performance information. The output of the display command goes to the operator console.

The syntax of the **DISPLAY** command for LDAP is:

```
f dssrv,display report
```

Where *report* is the name of an LDAP server report, which is shown below.

The LDAP server provides a **RESET** operator modify command that is used to reset the statistics that are shown in some of the reports.

The syntax of the **RESET** command for LDAP is:

```
f dssrv,reset report
```

Where *report* is the name of an LDAP server report, which is shown below.

The following is a description of each of the LDAP server reports. Where applicable, the reset processing is also described.

AUDIT

This option shows the current audit settings. It shows if auditing is ON or OFF and what level of auditing is active for each event.

Following is sample output if you request an **AUDIT** report:

```
GLD1190I Audit status
Option     Setting
AUDIT      ON
ERROR      BIND
ALL        ADD CONNECT DELETE DISCONNECT MODIFY SEARCH
NONE       MODIFYDN
```

BACKENDS

This option shows the current state of all configured GDBM, LDBM, CDBM, SDBM, and TDBM backends.

The state is composed of two parts:

- The first part indicates whether the backend allows entries to be updated (READWRITE) or is read only (READONLY).
- The second part indicates the backend processes requests (AVAILABLE), is not available because DB2 is down (UNAVAILABLE), is disabled because it is not able to load its directory (DISABLED), or the database files cannot be written because of a file system error (UNWRITABLE).

The state is set to FAILED CONFIGURATION if the backend fails to start.

Following is sample output if you request a **BACKENDS** report:

```
GLD1224I Backend status
Backend                State
mycdbm                 READWRITE,AVAILABLE
GDBM-001               READWRITE,AVAILABLE
TDBM-002               READWRITE,UNAVAILABLE
myTDBM                 FAILED CONFIGURATION
Dept1LDBM              READONLY,AVAILABLE
Dept2LDBM              READWRITE,DISABLED
SDBM-003               READWRITE,AVAILABLE
Dept3LDBM              READONLY,UNWRITABLE
```

DEBUG

This option shows the current **DEBUG** setting.

Following is sample output if you request a **DEBUG** report with debug set to THREAD+PERF+TDBM:

```
GLD1226I Debug settings
THREAD
PERFORMANCE
TDBM
```

LEVEL

This option shows the current server version, including service level and **serverCompatLevel** setting.

Following is sample output if you request a **LEVEL** report:

```
GLD1001I LDAP server version 4.2, Service level OAxxxxx.
GLD1315I LDAP server compatibility level 7.
```

LOCKS

This option shows current server lock contention. The output indicates how often there has been contention on a lock and the average duration of

contention (in microseconds). If a lock is held, the output indicates what function and thread are holding the lock and how long the lock has been held.

Following is sample output if you request a **LOCKS** report:

```
GLD1126I Server lock statistics
Lock                                      Shared             Exclusive
Activity Log                             0 0.000000           0 0.000000
SRV WLM                                  0 0.000000           0 0.000000
SRV Operations Monitor                   0 0.000000           0 0.000000
Schema - DN Cache                        0 0.000000           0 0.000000
DN Cache                                26 0.001125          60 0.000584
LDBM Replicate                           0 0.000000           0 0.000000
LDBM - Filter Cache                      0 0.000000           0 0.000000
LDBM Filter Cache                        0 0.000000           0 0.000000
LDBM Database                           65 0.053177          69 0.113185
TDBM - DN to EID Cache                   0 0.000000           0 0.000000
TDBM - Entry Cache                       0 0.000000           0 0.000000
TDBM - Filter Cache                      0 0.000000           0 0.000000
TDBM - Entry Owner Cache                 0 0.000000           0 0.000000
TDBM - ACL Source Cache                  0 0.000000           0 0.000000
TDBM Replication                         0 0.000000           0 0.000000
TDBM Group Cache                       674 0.478458          21 0.496418
held by tdbm_build_group_cache for 1 seconds by thread=10
TDBM ACL Cache                           0 0.000000           0 0.000000
TDBM Filter Cache                        0 0.000000           0 0.000000
TDBM Alias Cache                         0 0.000000           0 0.000000
TDBM Referral Cache                      0 0.000000           0 0.000000
TDBM DN Cache                            0 0.000000           0 0.000000
TDBM Entry Cache                         1 0.001386          17 0.001209
TDBM EID Assignment                      0 0.000000           0 0.000000
TDBM Schema Lookup                       0 0.000000           0 0.000000
TDBM - DN to EID Cache                   0 0.000000           0 0.000000
TDBM - Entry Cache                       0 0.000000           0 0.000000
TDBM - Filter Cache                      0 0.000000           0 0.000000
TDBM - Entry Owner Cache                 0 0.000000           0 0.000000
TDBM - ACL Source Cache                  0 0.000000           0 0.000000
TDBM Replication                         0 0.000000           0 0.000000
TDBM Group Cache                         0 0.000000           0 0.000000
TDBM ACL Cache                           0 0.000000           0 0.000000
TDBM Filter Cache                        0 0.000000           0 0.000000
TDBM Alias Cache                         0 0.000000           0 0.000000
TDBM Referral Cache                      0 0.000000           0 0.000000
TDBM DN Cache                            0 0.000000           0 0.000000
TDBM Entry Cache                         0 0.000000           0 0.000000
TDBM EID Assignment                      0 0.000000           3 0.006501
TDBM Schema Lookup                       0 0.000000           0 0.000000
Schema                                   0 0.000000           0 0.000000
Message                                  0 0.000000           0 0.000000
Monitor                                  0 0.000000           0 0.000000
```

The lock statistics can be reset to 0 by using the **RESET LOCKS** command.

LOG This option shows the current activity log settings.

Following is sample output if you request a **LOG** report:

```
GLD1290I Activity log status
Option:             Setting
OPERATIONS          ALLOPS
TIME                NOTIME
MESSAGES            NOMSGS
MICROSECONDS        MICRO
FILTER              (|(ibm-filterIP=192.1.1.*) (ibm-filterIP=193.1.*))
```

MAINTMODE

This option shows if maintenance mode is ON or OFF for the LDAP server.

Following is sample output if you request a **MAINTMODE** report:

```
GLD1225I Maintenance Mode status
Maintenance mode  OFF
```

MONITOR

This option shows the monitor report. This report provides operational statistics for both the server and for each backend that successfully started, including information about the number of specific operations that are targeted at the LDAP server or to a given backend. This information can be used to better tune backend caches for improved performance.

The operations monitor statistics are not returned in this report. These statistics can be retrieved by performing an LDAP search of the operations monitor. See "Monitoring performance with cn=monitor" on page 612 for more information.

Following is sample output if you request a **MONITOR** report:

```
            Monitor Statistics
            ------------------

Server Version:      z/OS Version 2 Release 1 IBM Tivoli Directory
                     Server LDAP Server
Current Time:        Thu May 23 17:53:37.145092 2013
Start Time:          Thu May 23 14:28:48.835516 2013
Last Reset Time:     Thu May 23 14:28:48.835516 2013
Number of Resets:    0

Server Totals:
--------------

Description                 Count
---------------------------------
Config Max Connections      24982
System Max Connections      25000
Total Connections           31927
Current Connections             0
MaxReached Connections          2
Timed Out Connections           0
Timed Out Result Sets           0
Paged Entries Cached            0
Paged Searches                844
Sorted Searches              3423
Group Gathering Events      31645
Search Entries Sent        332905
Message Bytes Sent       83801792
Search References Sent          0
Search Pages Sent            1688

Operation    Requested  Completed
-----------  ---------  ---------
Abandons             0          0
Adds              7013       7013
AllOps          113783     113783
Binds            31927      31927
Compares           133        133
Deletes           7013       7013
ExtOps             333        333
Modifies         14058      14058
ModifyDNs         1290       1290
Searches         17093      17093
Unbinds          31926      31926
Unknown              0          0

Backend Statistics: cdbm
------------------
Naming Context:     CN=CONFIGURATION
Naming Context:     CN=IBMPOLICIES

Description                 Count
---------------------------------
Group Gathering Events          0
Search Entries Sent             0
Message Bytes Sent             14
Search References Sent          0

Operation    Requested  Completed
-----------  ---------  ---------
Abandons             0          0
```

```
Adds                0          0
AllOps              2          2
Binds               0          0
Deletes             0          0
ExtOps              0          0
Modifies            2          2
ModifyDNs           0          0
Searches            0          0
Unbinds             0          0
Unknown             0          0
                                                          Pct  Refresh  Refresh
Cache Name     Size  Entries     Hits   Misses    Hit    Count  AvgEnts
----------  -------  -------  -------  -------  -----  -------  -------
Filter         5000        2        0       13   0.0%        8        1

Filter Bypass Limit: 100

Backend Statistics: TDBM-0001
-----------------
Naming Context:     DC=YOURCOMPANY2,DC=COM
Naming Context:     C=TDBM
Naming Context:     CN=MOVER
Naming Context:     CN=MOVING

Description                 Count
---------------------------------
Group Gathering Events      24101
Search Entries Sent        167330
Message Bytes Sent       42446210
Search References Sent           0

Operation    Requested  Completed
-----------  ---------  ---------
Abandons             0          0
Adds              4980       4980
AllOps           32623      32623
Binds                0          0
Compares           133        133
Deletes           4980       4980
ExtOps               0          0
Modifies         10098      10098
ModifyDNs          874        874
Searches         11558      11558
Unbinds              0          0
Unknown              0          0

                                                          Pct  Refresh  Refresh
Cache Name     Size  Entries     Hits   Misses    Hit    Count  AvgEnts
----------  -------  -------  -------  -------  -----  -------  -------
ACLSource       100        0   239594    12715   95.0%        0        0
DnToEID        1000        0    30724    11957   72.0%        0        0
Entry          5000        0        0   253043    0.0%        0        0
EntryOwner      100        0   252060      249   99.9%        0        0

Filter Bypass Limit: 100

Backend Statistics: LDBM-0002
-----------------
Naming Context:     C=DE
Naming Context:     DC=YOURCOMPANY,DC=COM
Naming Context:     C=CA
Naming Context:     C=AU
Naming Context:     C=LDBM

Description                 Count
---------------------------------
Group Gathering Events       7544
Search Entries Sent         81472
Message Bytes Sent       20424224
Search References Sent           0

Operation    Requested  Completed
-----------  ---------  ---------
Abandons             0          0
Adds              2033       2033
AllOps           13968      13968
Binds                0          0
Compares             0          0
Deletes           2033       2033
```

```
ExtOps              0          0
Modifies         3956       3956
ModifyDNs         416        416
Searches         5530       5530
Unbinds             0          0
Unknown             0          0

                                                          Pct  Refresh  Refresh
Cache Name     Size  Entries     Hits   Misses    Hit    Count  AvgEnts
----------  -------  -------  -------  -------  -----  -------  -------
Filter         5000        0     5529        1  100.0%    8438        0

Filter Bypass Limit: 100

Backend Statistics: SDBM-0003
-----------------
Naming Context:     CN=MYRACF

Description                 Count
---------------------------------
Search Entries Sent             0
Message Bytes Sent              0
Search References Sent          0

Operation    Requested  Completed
-----------  ---------  ---------

Abandons             0          0
Adds                 0          0
AllOps               0          0
Binds                0          0
Compares             0          0
Deletes              0          0
ExtOps               0          0
Modifies             0          0
ModifyDNs            0          0
Searches             0          0
Unbinds              0          0

Backend Statistics: Monitor
-----------------
Naming Context:     CN=MONITOR

Description                 Count
---------------------------------
Search Entries Sent             2
Message Bytes Sent           2700
Search References Sent          0

Operation    Requested  Completed
-----------  ---------  ---------
Abandons             0          0
Adds                 0          0
AllOps               2          2
Binds                0          0
Compares             0          0
Deletes              0          0
ExtOps               0          0
Modifies             0          0
ModifyDNs            0          0
Searches             2          2
Unbinds              0          0
Unknown              0          0

Backend Statistics: Schema
-----------------
Naming Context:     CN=SCHEMA

Description                 Count
---------------------------------
Search Entries Sent             0
Message Bytes Sent             28
Search References Sent          0

Operation    Requested  Completed
-----------  ---------  ---------
Abandons             0          0
Adds                 0          0
AllOps               2          2
```

```
Binds                0          0
Compares             0          0
Deletes              0          0
ExtOps               0          0
Modifies             2          2
ModifyDNs            0          0
Searches             0          0
Unbinds              0          0
Unknown              0          0

                                                      Pct  Refresh  Refresh
Cache Name     Size  Entries     Hits   Misses    Hit    Count  AvgEnts
----------  -------  -------  -------  -------  -----  -------  -------
Dn             1000       84   300676      117  100.0%       2       18

Backend Statistics: RootDSE
------------------

Description                 Count
---------------------------------
Search Entries Sent             3
Message Bytes Sent           4039
Search References Sent          0

Operation    Requested  Completed
-----------  ---------  ---------
Abandons             0          0
Adds                 0          0
AllOps               3          3
Binds                0          0
Compares             0          0
Deletes              0          0
ExtOps               0          0
Modifies             0          0
ModifyDNs            0          0
Searches             3          3
Unbinds              0          0
Unknown              0          0
```

Note: Some statistics might 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)
```

The monitor statistics can be reset by using the **RESET MONITOR** command, as described below. The reset statistics are reflected in future **DISPLAY MONITOR** reports and **cn=monitor** searches. **RESET MONITOR** has no effect on the values reported in the Activity Log. When monitor 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, Group Gathering Events, Timed Out Connections, Timed Out Result Sets, Paged Searches, Sorted Searches, Search Entries Sent, Search Pages 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.

The **RESET MONITOR** command also resets operations monitor statistics. See “Monitoring performance with cn=monitor” on page 612 for more information.

Resetting the monitor statistics has no effect on the Cache **Size** for each cache, nor on the **Filter Bypass Limit**, because these are configured values. It also has no effect on the Cache **Entries** for each cache, because the contents of the caches are not altered by a reset of statistics. In addition, **Paged Entries Cached** are not affected by a monitor reset.

Some caches might get invalidated and refreshed because of 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.

The same statistics in this report can also be obtained by using the **ldapsearch** utility to do a subtree search of the **cn=monitor** entry.

See "Monitoring performance with cn=monitor" on page 612 for more information about searching by using **cn=monitor**.

NETWORK

This option shows the current state of all configured network interfaces. If the interface IP address is `0.0.0.0`, it indicates that the LDAP server is listening on the **INADDR_ANY** interface. If the interface IP address is `::`, it indicates that the LDAP server is listening on the **in6addr_any** interface.

Following is sample output if you request a **NETWORK** report:

```
GLD1084I Network interface status
Interface                                Port  Status
127.0.0.1                                3389  ACTIVE
127.0.0.1                                 389  ACTIVE
127.0.0.1                                1492  ACTIVE
9.12.47.95                               3389  ACTIVE
9.12.47.95                                389  ACTIVE
9.12.47.95                               1492  ACTIVE
0.0.0.0                                   489  ACTIVE
::                                        589  ACTIVE
```

REPLICAS

This option behaves differently if basic replication or advanced replication are enabled.

If basic replication is configured, this option shows the current state of each replication replica. The report indicates how many updates are currently in each replication queue, how many updates have been successfully replicated, and how many failed replication attempts have been set aside. This information can be used to determine if replication has stalled for a certain replica, and, if it has, can determine how many updates are left to process for that replica.

Following is sample output if you request a **REPLICAS** report and are using basic replication:

```
GLD1163I Replication status
Backend   Replica              Queued/total/Set Aside
TDBM1     ldap2.ibm.com:389     10    /134    /0
TDBM1     ldap1.ibm.com:389     0     /134    /0
```

If advanced replication is configured, this option gives a view of the advanced replication topology configured. This includes basic state information for each replication context and replication agreement. Operational attributes retrievable from replication context and agreement entries can provide more detailed information about the state of advanced replication on the server.

Following is the sample output if you request a **REPLICAS** report and are using advanced replication:

On the system console:

```
GLD1296I Display Replica completed.
```

In LDAP server's job log:

```
090827 12:38:31.577855 GLD8502I Replication context CN=IBMPOLICIES
 replica type: SUPPLIER
 context state: NORMAL
 ibm-serverID: 9AB14000-7BF1-1807-9F7F-4020940EC8E0
 ibm-subentryDN: CN=MASTER,IBM-REPLICAGROUP=DEFAULT,CN=IBMPOLICIES
 agreements: 1 agreements defined
 referrals: NULL

090827 12:38:31.578651 GLD8506I Replication agreement
CN=ZFORWARDER,CN=MASTER,IBM-REPLICAGROUP=DEFAULT,CN=IBMPOLICIES
 context DN: CN=IBMPOLICIES
 state: on hold
 ibm-replicaURL: ldap://localhost:2492
 ibm-replicaCredentialsDN: CN=ZFORWARDER SIMPLE,CN=REPLICATION,O=SAMPLELDBM
 ibm-replicationFilterDN: NULL
 ibm-replicaScheduleDN: NULL
 bind-info: bind_dn "cn=admin2", method SIMPLE
 connection-status: connected=BOUND, connection type=CLEAR
 agreement-status: last changeID sent=8, errors logged=1, on hold=TRUE
 pending change count: 4
```

THREADS

This option shows the current activity statistics for the communication threads in the LDAP server. These threads are used to handle the network communications between the LDAP server and clients and to process LDAP operations that are requested by the clients.

The values that are returned for a **THREADS** report are:

commThreads defined
: number of communication threads. This is the value of the **commThreads** option in the LDAP server configuration file.

current active commThreads
: number of communication threads currently active

max concurrent commThreads
: largest number of communication threads that have been active at the same time

average thread activity
: percentage of time that the communication threads have been active

current active LDAP operations
: number of communication threads that are currently processing LDAP operations

max concurrent LDAP operations
: largest number of communication threads that have been processing LDAP operations at the same time

LDAP operations per second
: average number of LDAP operations processed per second

client network connections per second
: average number of LDAP client connections processed per second

Following is sample output if you request a **THREADS** report:

```
GLD1109I Server activity statistics
20 commThreads defined
2 current active commThreads
```

```
16 max concurrent commThreads
20.59% average thread activity
2 current active LDAP operations
15 max concurrent LDAP operations
39.66 LDAP operations per second
13.22 client network connections per second
```

These statistics, except for the 'commThreads defined' value, are reset to 0 using the **RESET THREADS** command.

XCF This option shows the current state of all servers in the sysplex group. This report indicates which server is the group owner, what servers are currently active in the group, and which server is a transition server. For more transition server information, see "Updating LDAP configurations settings in a sysplex without server outage" on page 210.

Following is sample output if you request an **XCF** report:

```
GLD1135I Sysplex status
Server                          System        Status
DCEIMGVV0066-BF31AD8E-2EFB4F4E  DCEIMGVV0066  ACTIVE/REPLICA/TRANSITION
DCEIMGVV004E-BF31AD83-ACE95202  DCEIMGVV004E  ACTIVE/OWNER
```

The System name for an LDAP server in a sysplex group is displayed in message GLD1146I when the server is started.

Size limitations

If any of the above reports exceeds 256 lines of output, the report is truncated and the following message is displayed as the last line of the output:

```
GLD1086I Maximum number of lines displayed.
```

If the **MONITOR** report is truncated, the complete output can be found in the LDAP server job log. The same statistics can also be obtained by using **ldapsearch** to do a subtree search of the **cn=monitor** entry. See "Monitoring performance with cn=monitor" on page 612 for more information about searching **cn=monitor**.

Activity logging

The LDAP server supports logging client activity in an activity log file. The activity log file can be analyzed for LDAP server load analysis to determine the client operations that are handled by the server. The activity log records can contain information about operations that are handled by the server, client IP addresses, messages that are generated by the server, and hourly activity summary statistics. The LDAP server activity logging support has a number of features that allow the LDAP root administrator to customize the client activity to be logged.

The available versions of activity logging are version 0 and version 1. Activity log version 1 is the enhanced activity logging version that includes more features to help further debugging LDAP issues and inappropriate user operations of the LDAP server besides all features in version 0. For example, the addition of logging attributes in the add and modify requests allows you to detect when and who modified the critical attributes. The connect and disconnect records allow the detection of software configuration errors or denial of service attacks. The abandon record shows when a request is abandoned and the `msgid` of the request that is abandoned.

The features for version 0 include:

- Logging the start or end of a client operation.

- Logging only client update operations (add, delete, modify, extended operations, and modifydn).
- Logging all client operations (add, bind, compare, delete, extended operations, modify, modifydn, search, and unbind).
- Logging messages that are generated by the server.
- Logging hourly client activity summary statistics.
- Logging only requests from certain client IP addresses.
- Activity log file archiving or rollover that copies the current activity log file to another location for load analysis.

The features for version 1 include:

- Logging the start or end of a client operation.
- Logging only client update operations (add, delete, modify, extended operations, and modifydn).
- Logging all client operations (add, bind, compare, delete, extended operations, modify, modifydn, search, and unbind).
- Logging messages that are generated by the server.
- Logging hourly client activity summary statistics.
- Logging only requests from certain client IP addresses.
- Activity log file archiving or rollover that copies the current activity log file to another location for load analysis.
- A configuration keyword that enables the new version, thus users that want the previous behavior are not effected.
- Logging the start and end of a connection.
- Logging abandon requests.
- Adding the `msgid` to requests that can be abandoned so that abandon requests can be matched to the requests they affect. The requests include: add, bind, compare, delete, exop, modify, rename, and search.
- Logging the attribute names in the add request records.
- Logging the attribute names and an indication of the attribute being added, deleted, or replaced in the modify request records.
- Logging requests unknown to the activity log.

Start and end connection logging

The following is an example of version 1 merged activity log records containing connection start and connection end messages:

```
Thu Oct 8 08:37:59 2009 mergedRecord Connect: connid = 1374, clientIP = 1.2.3.4,
 listen = ldap://[fe00::f4f7:0:0:7442:7510]:389
Thu Oct 8 08:50:59 2009 mergedRecord Disconnect: connid = 1374, clientIP = 1.2.3.4,
 bind = 'cn=john', cause = 1
```

The following is an example of version 1 non-merged activity log records containing connection start and connection end messages:

```
Thu Oct 8 08:43:43.424715 2009 Connect: connid = 1, listen =
 ldap://[fe00::f4f7:0:0:7442:7510]:389, IP = 1.2.3.4,
Thu Oct 8 08:49:45.424716 2009 Disconnect: connid = 1, DN = 'cn=john', cause = 1,
 IP = 1.2.3.4
```

Abandon request

The following is an example of a version 1 merged activity log record for an abandon request:

```
Thu Oct 8 08:37:59 2009 mergedRecord Abandon : connid = 1374,
 clientIP = 9.12.47.67, time = 711usec, msgid = 123, targetmsgid = 456
```

The following is an example of version 1 non-merged activity log records for an abandon request:

```
Thu Oct 8 08:43:43.424715 2009 Abandon: connid = 1, msgid = 123, targetmsgid = 456,
 IP = 1.2.3.4
Thu Oct 8 08:49:45.424716 2009 End Abandon: connid = 1, msgid = 123,
 targetmsgid = 456, IP = 1.2.3.4
```

Attribute names in add request records

Attribute names is a string of attribute names that are separated by spaces. The following is an example of a version 1 merged activity log record for an add request:

```
Thu Oct 8 08:37:59 2009 mergedRecord Add: connid = 1374, clientIP = 1.2.3.4,
 bind = 'cn=john,ou=zos,o=ibm,c=us', rc = 0, time = 49268usec,
 controls = 1.3.6.1.4.1.42.2.27.8.5.1, target = ou=zos,o=ibm,c=us, msgid = 789,
 attrs = cn objectclass description, rsn = NA
```

The following is an example of a version 1 non-merged activity log record for an add request:

```
Thu Oct 8 08:43:43.424715 2009 Add: connid = 1, DN = 'cn=john,ou=zos,o=ibm,c=us', msgid = 789,
 IP = 1.2.3.4
Thu Oct 8 08:49:45.424716 2009 End Add: connid = 1, DN = cn=john,ou=zos,o=ibm,c=us, rc = 0,
 msgid = 789, attrs = cn objectclass description, IP = 1.2.3.4
```

Attribute names in modify request records

Version 1 of activity logging provides attribute names and an indication of the attribute being added, deleted, or replaced in the modify request records.

Attribute names in a modify request record is a string with an indication of:
the action taken separated by spaces, ended by a comma.
"-" indicates that the attribute is deleted.
"+" indicates that the attribute is added.
"!" indicates that the attribute is replaced.

The following is an example of a version 1 merged activity log record for a modify request:

```
Thu Oct 8 08:37:59 2009 mergedRecord Modify: connid = 1374, clientIP = 1.2.3.4,
 bind = 'cn=john,ou=zos,o=ibm,c=us', rc = 0, time = 49268usec,
 controls = 1.3.6.1.4.1.42.2.27.8.5.1, target = 'ou=zos,o=ibm,c=us', msgid = 789,
 attrs = -telephone +mobilephone !socsecuritynumber, rsn = NA
```

The following is an example of a version 1 non-merged activity log record for a modify request:

```
Thu Oct 8 08:43:43.424715 2009 Modify: connid = 1, DN = 'cn=john,ou=zos,o=ibm,c=us', msgid = 789,
 IP = 1.2.3.4
Thu Oct 8 08:49:45.424716 2009 End Modify: connid = 1,
 DN = 'cn=john,ou=zos,o=ibm,c=us', rc = 0, msgid = 789, attrs = -telephone +mobilephone
 !socsecuritynumber, IP = 1.2.3.4
```

Logging requests unknown to activity logging

The following is an example of a version 1 merged activity log record for an unknown request:

```
Thu Oct 8 08:37:59.424715 2009 mergedRecord Unknown: type = 25, connid = 1, clientIP = 9.12.47.67,
 bind = 'cn=Admin', rc = 0, time = 711usec, controls = , msgid = 2, rsn = NA
```

The following is an example of a version 1 non-merged activity log record for an unknown request:

```
Tue Oct 9 02:16:23.822710 2009 Unknown: type = 25, connid = 1, msgid = 2, IP = 9.12.47.67
Tue Oct  9 02:16:23.822789 2009 End Unknown: type = 25, connid = 1,  msgid = 2, IP = 9.12.47.67
```

Configuring the activity log support

The **logfile** configuration option in the global section of the LDAP server configuration file specifies the location of the z/OS UNIX System Services file or data set where activity log records are written. If a z/OS UNIX System Services file is specified for the **logfile** configuration option, a fully qualified name must be specified. You should use a sequential data set if you want to use data set for the activity log file. If you must use a partitioned data set, make sure that the record length of the data set is large enough to contain all possible activity log records. Otherwise, the LDAP server fails to write the specific activity log record and the activity log is disabled.

The following is an example of a **logfile** configuration option specifying a z/OS UNIX System Services file:

```
logfile /etc/ldap/ldap.activity.log
```

If the **logfile** configuration option is not specified, the default location of the activity log file is **/etc/ldap/gldlog.output**. A partitioned or sequential data set can be specified in the **logfile** configuration option by using a DDNAME JCL card or as a specific data set. If a data set is to be used for the activity log, create it before it is used because the LDAP server uses system defaults to create the data set and they might not be correct for the activity log. If a GDG (Generated Data Group) data set is used for the activity log, the GDG base must be specified for the **logfile** configuration option, and the base must be created before the server is started. The sample JCL in Figure 7 creates a base GDG named, `GLD.GDGBASE`, that might contain up to three generated data sets.

```
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  DEFINE GDG(  -
     NAME(GLD.GDGBASE)     -
     LIMIT(3)  -
    NOEMPTY  -
    SCRATCH)
/*
//
```

Figure 7. GDG JCL example

The LDAP server ignores any model that is associated with the activity log's GDG when creating new generation data sets, therefore, there is no need to create the model GDG. The LDAP server uses the Storage Management Subsystem (SMS) defaults for that high-level qualifier when it creates the generation data sets. See "Configuration file options" on page 90 for more information about the **logfile** configuration option.

The following are examples of specifying data sets for the activity log file:

```
logfile //DD:LOGOUT
logfile //'GLD.ACTLOG'
logfile //'GLD.MYLOG(OUT)'
logfile //'GLD.GDGBASE'
```

The LDAP server supports automatic activity log file rollover or archiving based on the time of day or the size of the log file. The activity log archiving is only supported when the log file is a z/OS UNIX System Services file or a GDG (Generated Data Group) data set. When the **logFileRolloverTOD** configuration option has a value between 00:00 and 23:59, it indicates the time each day when the current activity log file is archived. See "Configuration file options" on page 90 for more information about the **logFileRolloverTOD** configuration option. When the **logFileRolloverSize** configuration option has a nonzero size, it indicates the size in bytes, megabytes, kilobytes, or gigabytes that the activity log file is required to have before it is archived or rolled over. See "Configuration file options" on page 90 for more information about the **logFileRolloverSize** configuration option. When the activity log file reaches one of these thresholds or is manually rolled over with the **LOG** server operator modify command, the following occurs if the log file is a z/OS UNIX System Services file:

- The current activity log file is renamed with the current Coordinated Universal Time stamp appended to the end of the file name.
- If the **logFileRolloverDirectory** configuration option is specified, the archived log file is moved to that directory, or else the archived activity log file is left in the same directory as the current activity log file. Activity log archiving or rollover is only supported when the underlying z/OS UNIX System Services file system is the same between the directory in the specified **logfile** configuration option and the directory that is specified in the **logFileRolloverDirectory**. See "Configuration file options" on page 90 for more information about the **logFileRolloverDirectory** configuration option.
- When archiving is complete, the server opens a new activity log file with the name specified in the **logfile** configuration option.

When archiving using data sets, specify the same Generated Data Group (GDG) base name for both the **logfile** and **logFileRolloverDirectory** configuration options. When logging is first activated, a new data set generation in the GDG is created and log records are written to that data set. Each time rollover or archiving occurs, the current activity log file is closed, and a new data set generation is used.

Note: When archiving using a GDG, no more than 255 new data set generations can be created during the life of the server. If the limit is exceeded, the old generations are reused and the log data is overwritten. To avoid exceeding this limit, you should plan the rollover strategy and data set capacity accordingly. Alternatives include:

- Archive once per day using **logFileRolloverTOD**, if your server is restarted more often than every 255 days. Ensure the data set allocations accommodate a full day's worth of activity log records.
- Archive based on data set size, make the **logFileRolloverSize** large enough such that 255 generations do not occur before the server is restarted. Ensure that the data set allocations accommodate the size that is chosen.
- Consider logging and archiving using z/OS UNIX System Services files instead because they are not constrained by this limitation.

Activity log file rollover or archiving is not supported when the **logfile** or **logFileRolloverDirectory** configuration options specify a partitioned data set, sequential data set, or a DD card.

When the activity log is configured to log client operations and is active, the LDAP server logs all client operations from all IP addresses. However, the activity log can be configured to include or exclude client operations from certain IP addresses by using the **logFileFilter** configuration option or the **LOG** operator modify

command. See "Configuration file options" on page 90 for more information about the **logFileFilter** configuration option. An IETF RFC 2254 (RFC 2254: *The String Representation of LDAP Search Filters*) compliant LDAP search filter can be specified in the **logFileFilter** configuration option or the **LOG** operator modify command using only the **ibm-filterIP** attribute type in the filters. The following example **logFileFilter** configuration option specifies to only log client operations from IP addresses starting with `1.2.3* or 1.2.4*`.

```
logFileFilter (|(ibm-filterIP=1.2.3*)(ibm-filterIP=1.2.4*))
```

The default data collection setting is to collect no data in the activity log. The default is modified by environment variables or through the usage of the LDAP server **LOG** operator modify command. The environment variables are read as the server starts when the **LOG** operator modify command can be specified once the server has started. The **logFileFilter** configuration option can also be updated with the **LOG** operator modify command.

To enable the version 1 features of the activity log, specify the **logFileVersion** configuration option in the configuration file, with the value of 1. For more information about how to configure the version, see "Configuration file options" on page 90 for the **logFileVersion** configuration option.

An operator modify command can be sent to the LDAP server from the SDSF or the operator's console. Note if the command is entered from SDSF, it must be preceded by a slash (/). The format of the LDAP server **LOG** operator modify command is:

```
F DSSRV, LOG { WRITEOPS | ALLOPS | SUMMARY | TIME | NOTIME | MERGEDRECORD | MSGS | NOMSGS | FLUSH | STOP | ROLLOVER | FILTER,filter }
```

where one of the options in the { } must be specified. No more than one option in the { } is allowed to be specified at a time on the **LOG** operator modify command and the { } should not be specified as part of the command.

If the activity log must be restarted with logging the beginning and end of all operations, specify the following **LOG** operator modify commands:

```
F DSSRV, LOG ALLOPS
```

then,

```
F DSSRV, LOG TIME
```

The **logFileOps** configuration option and the **LOG** operator modify command can be used to control which operations generate log records. Before specifying an operation setting, no operation logging is performed. See the **logFileOps** configuration option at "Configuration file options" on page 90 for more information.

Summary records are created on an hourly basis, when roll over or archiving occurs, or when an LDAP server **LOG** operator modify command is processed.

Note: If activity logging is not active, summary records are not created.

The summary log records contain information about the operations that the server has processed.

Following is an example of the summary log records:

```
Tue Jun 15 14:26:43.785091 2010 total operations started = 113780
Tue Jun 15 14:26:43.785165 2010 total operations completed = 113780
Tue Jun 15 14:26:43.785185 2010 total binds completed = 31926
```

```
Tue Jun 15 14:26:43.785202 2010 total unbinds completed = 31925
Tue Jun 15 14:26:43.785221 2010 total searches completed = 17092
Tue Jun 15 14:26:43.785238 2010 total modifies completed = 14058
Tue Jun 15 14:26:43.785254 2010 total adds completed = 7013
Tue Jun 15 14:26:43.785270 2010 total deletes completed = 7013
Tue Jun 15 14:26:43.785286 2010 total modifydns completed = 1290
Tue Jun 15 14:26:43.785302 2010 total compares completed = 133
Tue Jun 15 14:26:43.785318 2010 total abandons completed = 0
Tue Jun 15 14:26:43.785334 2010 total extendedops completed = 3330
Tue Jun 15 14:26:43.785349 2010 total unknown completed = 0
Tue Jun 15 14:26:43.785355 2010 total group gatherings completed = 98876
Tue Jun 15 14:26:43.785366 2010 total search entries sent = 332904
Tue Jun 15 14:26:43.785381 2010 total bytes sent = 83800442
Tue Jun 15 14:26:43.785396 2010 total search references sent = 0
Tue Jun 15 14:26:43.785411 2010 total search pages sent = 1688
Tue Jun 15 14:26:43.785427 2010 total paged searches completed = 844
Tue Jun 15 14:26:43.785445 2010 total sorted searches completed = 3423
Tue Jun 15 14:26:43.785459 2010 total connections processed = 31926
Tue Jun 15 14:26:43.785476 2010 current connections = 0
Tue Jun 15 14:26:43.785491 2010 connection high water mark = 2
Tue Jun 15 14:26:43.785506 2010 connections timed out = 0
Tue Jun 15 14:26:43.785521 2010 paged result sets timed out = 0
```

The **logFileRecordType** configuration option and the **LOG** operator modify command can be used to control when log records are generated. See the **logFileRecordType** configuration option at "Configuration file options" on page 90 for more information.

The following are examples of activity log records containing start and end records:

```
Thu Oct  8 07:41:08 2009 Bind SIMPLE: connid = A, DN = cn=john,ou=zos,o=ibm,c=us, IP = 1.2.3.4
Thu Oct  8 07:41:08 2009 End Bind SIMPLE: connid = A, DN = cn=john,ou=zos,o=ibm,c=us, safid = , rc = 0,
 IP = 1.2.3.4, policyUpdated = T
Thu Oct 8 07:41:08 2009 Search: connid = A, base = ou=zos,o=ibm,c=us, filter = (objectclass=*), scope = 2,
 attrs = , IP = 1.2.3.4, searchFlags = 0
Thu Oct  8 07:41:08 2009 End Search: connid = A, base = ou=zos,o=ibm,c=us, filter = (objectclass=*), scope = 2,
 count = 4, rc = 0, IP = 1.2.3.4, searchFlags = 0
Thu Oct  8 07:41:08 2009 Unbind: connid = A, DN = cn=john,ou=zos,o=ibm,c=us, IP = 1.2.3.4
Thu Oct  8 07:41:08 2009 End Unbind: connid = A, DN = cn=john,ou=zos,o=ibm,c=us, IP = 1.2.3.4
```

The following are examples of activity log records merged records:

```
Thu Oct  8 08:37:59 2009 mergedRecord Bind: connid = 1374, clientIP = 1.2.3.4, bind = cn=john,ou=zos,o=ibm,c=us,
 rc = 0, time = 49268usec, controls = 1.3.6.1.4.1.42.2.27.8.5.1, listen = ldap://
 [fe00::f4f7:0:0:7442:7510]:389, seclabel = , mech = SIMPLE, saf = , policyUpdated = T, rsn=NA
Thu Oct  8 08:37:59 2009 mergedRecord Search: connid = 1374, clientIP = 1.2.3.4, bind = cn=john,ou=zos,o=ibm,c=us,
 rc = 0, time = 1437usec, controls = , target = ou=zos,o=ibm,c=us, filter = (objectclass=*), scope = 2, attrs = ,
 count = 4, searchFlags = 0, rsn=NA
Thu Oct  8 08:37:59 2009 mergedRecord Unbind: connid = 1374, clientIP = 1.2.3.4, bind = cn=john,ou=zos,o=ibm,c=us,
 rc = 0, time = 6usec, controls = , listen = ldap://[fe00::f4f7:0:0:7442:7510]:389, seclabel = , saf = , rsn=NA
```

See Appendix F, "Activity log records," on page 735 for a description of the different fields that are present in activity log records.

Note: When the client connects to the LDAP server over the PC interface, the IP address is reported as 'PC'. If a request is handled by the cross-system facility (XCF), the IP address is reported as 'XCF'.

The **logFileMicroseconds** configuration option controls if all generated log records contain microseconds in their time stamps. This setting cannot be modified by a **LOG** operator modify command. The default does not include microseconds in the time stamps. See the **logFileMicroseconds** configuration option at "Configuration file options" on page 90 for more information.

The following are examples of activity log records containing microseconds:

```
Thu Oct  8 08:43:43.424715 2009 Bind SIMPLE: connid = 1, DN = cn=tom,ou=zos,o=ibm,c=us, IP = 1.2.3.4
Thu Oct  8 08:43:43.455011 2009 End Bind SIMPLE: connid = 1, DN =  cn=tom,ou=zos,o=ibm,c=us, safid = , rc = 0, IP = 1.2.3.4,
 policyUpdated = F
Thu Oct  8 08:43:43.466558 2009 Search: connid = 1, base = cn=tom,ou=zos,o=ibm,c=us, filter = (objectclass=*), scope = 2,
 attrs = , IP = 1.2.3.4, searchFlags = 0
Thu Oct  8 08:43:43.471836 2009 End Search: connid = 1, base = cn=tom,ou=zos,o=ibm,c=us, filter = (objectclass=*), scope = 2,
```

```
 count = 1, rc = 0, IP = 1.2.3.4, searchFlags = 0
Thu Oct  8 08:43:43.478053 2009 Unbind: connid = 1, DN = cn=tom,ou=zos,o=ibm,c=us, IP = 1.2.3.4
Thu Oct  8 08:43:43.478181 2009 End Unbind: connid = 1, DN = cn=tom,ou=zos,o=ibm,c=us, IP = 1.2.3.4
```

The **logFileMsgs** configuration option and the **LOG** operator modify command can be used to control if log records are generated when messages are created by the LDAP server. See the **logFileMsgs** configuration option at "Configuration file options" on page 90 for more information.

The following is an example of an activity log record containing a message:

```
Thu Oct  8 08:43:24.748429 2009 GLD1059I Listening for requests on 127.0.0.1 port 389.
```

The activity log filter that is specified in the **logFileFilter** configuration option can be updated by issuing a **LOG** operator modify command. The following command updates the server to only log client requests originating from IP address `1.2.4.5`.

```
F DSSRV, LOG FILTER,(IBM-FILTERIP=1.2.4.5)
```

The current activity log can be manually rolled over or archived using the process described above by issuing the following **LOG** operator modify command:

```
F DSSRV, LOG ROLLOVER
```

As the log records 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 issuing the following **LOG** operator modify command:

```
F DSSRV, LOG FLUSH
```

To have the server stop collecting activity data, issue the following **LOG** operator modify command:

```
F DSSRV, LOG STOP
```

Activity logging can be started again by specifying a **LOG** operator modify command with a new setting.

```
F DSSRV, LOG ALLOPS
```

The current activity log settings can be queried by issuing the following **DISPLAY** operator modify command:

```
F DSSRV,DISPLAY LOG

GLD1290I Activity log status
Option:          Setting
OPERATIONS       ALLOPS
TIME             MERGEDRECORD
MESSAGES         MSGS
MICROSECONDS     NOMICRO
FILTER           NONE
```

LDAP SMF auditing

The LDAP server can be configured to generate SMF type-83 subtype three audit records. The SMF type-83 log records containing LDAP events can be unloaded by using the RACF SMF data Unload utility for further analysis by auditing tools. These audit records contain information that is provided on LDAP client operation requests. The LDAP server can be configured to write audit records when the operation successfully completes, when the operation fails or for either case. SMF type-83 subtype 3 audit records are not created by the LDAP server when a request is handled by a plug-in. The LDAP server uses the **R_auditx** callable

service to write the record to SMF. See "Additional setup for generating audit records" on page 50 for setup information.

Auditing events

Auditing of LDAP operations can be set up by using the **audit** option in the LDAP server configuration file. See "audit {on | off | all,operations | error,operations | none,operations}" on page 93 for more information.

When the LDAP server is running, auditing can be turned on or off and the specifications of which operations are to be audited and their associated audit level can be changed by using the LDAP server **AUDIT** operator modify command. The format of the AUDIT operator modify command is:

```
f dssrv,audit on | off | all,operations | error,operations | none,operations
```

When auditing is on, an LDAP SMF type 83 subtype three audit record is generated for an operation if the operation is specified on an audit level and the operation result matches the audit level.

A separate **audit** configuration option or **AUDIT** operator modify command must be issued to turn auditing on or off and to set each audit level. Multiple operations can be specified for a level by putting a **+** between them on the **audit** option or **AUDIT** command, or by specifying multiple **audit** options or **AUDIT** commands with the same level.

Operations can be audited all the time or only when they fail. The following audit levels are supported:

all
An LDAP audit record is generated for the specified operations.

error
An LDAP audit record is generated for the specified operations when they fail.

none
An LDAP audit record is not generated for the specified operations.

The supported values for operations can be one or more of: **abandon, add, bind, compare, connect, delete, disconnect, exop, modify, modifydn, search, unbind**

If an operation is specified in more than one level, the last level is used for the operation. If an operation is not specified in any level, the level defaults to **none** for that operation. Turning off auditing does not change the setting for the audit levels. If auditing is later turned on, the audit levels remains as they last were.

For example, the following **AUDIT** operator modify commands turn auditing on for modify and search operation failures and for all bind operations. The other operations are not audited.

```
f dssrv,audit error,modify+search+bind
f dssrv,audit all,bind
f dssrv,audit on
```

The current audit settings can be displayed by using the following LDAP server **DISPLAY** operator modify command:

```
f dssrv,display audit
```

The results for the **AUDIT** operator modify commands issued above are:

```
GLD1190I Audit status
Option     Setting
AUDIT      ON
ERROR      MODIFY SEARCH
ALL        BIND
NONE
```

Working with audit records

The LDAP events are logged in an SMF data set as type 83 subtype 3 records. The audit record is a mixture of binary and EBCDIC data. The general format of SMF type 83 records is described in *z/OS Security Server RACF Macros and Interfaces* under the topic for SMF records, Record type 83: Security events.

The RACF SMF Unload utility can be used to reformat the LDAP SMF type 83 subtype 3 records for easier analysis. These audit records can be reformatted in the following different forms:

- A tabular format, suitable for import to a relational database manager.
- eXtensible Markup Language (XML) documents

See *z/OS Security Server RACF Auditor's Guide* for information about using the RACF SMF Unload utility. This document describes how the reformatted data can further be processed by DB2, DFSORT, and XML applications.

Three samples are shipped with the LDAP server to use with the RACF SMF Unload utility to reformat the LDAP audit records. The samples can be found in *GLDHLQ*.SGLDSAMP.

Two samples are used to create the tabular format. LDAPDBTB defines the DB2 table spaces and tables that are needed to contain the LDAP audit records and LDAPDBLD loads the formatted audit records into the tables.

A single sample is used to create the XML format. GLDLDPSC provides an XML schema to define the XML tag language that is used to reformat the LDAP audit records.

For the format and content of the LDAP SMF Audit records and the RACF SMF unload utility tabular description, see Appendix E, "SMF records," on page 721.

Monitoring LDAP server resources

The LDAP server monitors the basic resources that it uses to ensure that they are still available. If a resource becomes not available, the LDAP server can be configured to end or to operate without the resource until the resource becomes available.

Server backends and plug-ins during startup

When the LDAP server is started, the server processes the LDAP server configuration file and then initializes each of the backends or plug-ins that are configured. If an error is detected during initialization of a backend or plug-in, that backend or plug-in is not usable. Based on the **srvStartUpError** option in the LDAP server configuration file, the LDAP server shuts itself down or continues running with those backends or plug-ins that successfully start. After the problem encountered during backend or plug-in initialization is fixed, the LDAP server must be restarted to make that backend or plug-in available. See Chapter 8, "Customizing the LDAP server configuration," on page 83 for the description of

the **srvStartUpError** configuration option. The option does not apply to resource problems encountered after the LDAP server backends or plug-ins have started. The LDAP server response to those problems is described below.

DB2

The LDAP server uses DB2 to store the directory information for the TDBM and GDBM (when DB2-based) backends. If one of these backends is configured, the LDAP server attempts to connect to DB2 when the server is started. If DB2 is not available, the LDAP server, based on the **db2StartUpRetryLimit** and **db2StartUpRetryInterval** configuration options in the LDAP configuration file, can periodically try again to connect to DB2. If a DB2 connection cannot be established, the LDAP server can, based on the **srvStartUpError** configuration option, shut itself down or continue running without access to TDBM and GDBM (when DB2-based). See page "db2StartUpRetryInterval num-seconds" on page 97 for the descriptions of the **db2StartUpRetryLimit, db2StartUpRetryInterval**, and **srvStartUpError** configuration options.

When the LDAP server has connected to DB2 and a TDBM or DB2-based GDBM backend is running, the LDAP server monitors DB2 and detects when it shuts down. Based on the **db2Terminate** option in the LDAP server configuration file, the LDAP server shuts itself down or continues running without access to the TDBM and GDBM backends and reconnects to DB2 when it becomes available. If the LDAP server is configured to continue running when DB2 terminates, DB2 connection errors, such as -923 and -924 errors, might be reported as the LDAP server reacts to DB2 terminating. However, these messages do not affect the running of the server. See "db2Terminate {terminate | recover | restore}" on page 98 for the description of the **db2Terminate** configuration option.

Network communications

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 one of the network interfaces in use by the server has failed. Based on the **tcpTerminate** option in the LDAP server configuration file, the LDAP server can shut itself down or try to reestablish the failed interface. See "tcpTerminate {terminate | recover}" on page 134 for the description of the **tcpTerminate** configuration option.

The **tcpTerminate** option also controls the LDAP server response to a failure when initializing the SSL and Kerberos interfaces if they have been configured. The LDAP server can stop or continue processing but does not use the failed interface. After the problem is fixed, the LDAP server must be restarted to make that interface available.

Similarly, the **srvStartUpError** option also covers a failure when initializing the LDAP PC callable support interface if that has been configured. The LDAP server terminates or continues processing but does not use the PC support. After the problem is fixed, the LDAP server must be restarted to make the PC interface available. Note that only one LDAP server on a system can try to activate the PC interface. If another LDAP server has already tried to initialize (successfully or unsuccessfully) the PC interface, stop that server before starting this server with PC support configured.

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 when 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 might signify that the **maxConnections** option in the LDAP server 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 might 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. See "Activity logging" on page 193 for more information about activity logging.

Another way of controlling the number of clients concurrently connected to the LDAP server is by using the **idleConnectionTimeout** option in the LDAP server configuration file. This option controls the amount of time that the LDAP server allows a client connection to remain connected to the server when the client is not actively sending or receiving data on the connection. When the **idleConnectionTimeout** setting has been exceeded, the LDAP server severs the idle LDAP client connection, therefore, freeing the resource for a new client connection. See "idleConnectionTimeout num-seconds " on page 103 for more information.

File system

The LDAP server uses the z/OS UNIX System Services file system to store the directory information for the LDBM, CDBM, GDBM (file-based) backends. The LDAP server detects when file system errors occur, such as no space available or inability to write to required files or file directories. Based on the **fileTerminate** option in the LDAP server configuration file, the LDAP server shuts itself down or continues running with the affected LDBM, GDBM, or CDBM backend in read-only mode (updates to the directory are rejected). When the file system problem has been dealt with, the operator can use the LDAP server **BACKEND** operator modify command to change the LDBM, GDBM, or CDBM backend back to read/write mode. See Chapter 8, "Customizing the LDAP server configuration," on page 83 for the description of the **fileTerminate** configuration option.

LDAP server abnormal termination

The LDAP server registers itself with the z/OS Automatic Restart Management (ARM) service if the **armName** option is specified in the LDAP server configuration file. ARM can be set up to automatically restart the LDAP server if it stops unexpectedly. See "Configuration file options" on page 90 for the description of the **armName** configuration option. See *z/OS MVS Setting Up a Sysplex* for more information about ARM.

LDAP server operator commands

The following LDAP server commands can be entered by using the operator modify command. For example, if the LDAP server started task is DSSRV, the operator command is:

```
f dssrv,cmd-name cmd-options
```

The list below describes the supported LDAP server operator commands and their command options:

AUDIT *audit_controls*
: Turn LDAP server auditing on or off and control what server activities result in creating an audit record. See "LDAP SMF auditing" on page 200 for more information about controlling LDAP server usage of audit.

BACKEND *backendName*=**RDWR | READ**
: Change a specific backend to read/write (normal) mode or read-only mode. The LDAP server can place a file-based backend into read-only mode if the LDAP server cannot access its checkpoint or database files. After correcting the access problem, the operator can use this command to reset the backend to read/write mode.

COMMIT
: Force all the file-based backends to commit their changes to the database files by merging in the changes from the checkpoint files. The changes are removed from the checkpoint files. This can be done to prevent the checkpoint files from growing too large.

DEBUG *level* **| OUTPUT=MEMORY | OUTPUT=BOTH**
: Set the level of debugging. See "Dynamic debugging" on page 182 for more information about setting the debug levels. The command can also control where debug output is sent: to just the internal CTRACE table or to the CTRACE table and the normal debug output destination.

DISPLAY AUDIT | BACKENDS | DEBUG | LEVEL | LOCKS | LOG | MAINTMODE | MONITOR | NETWORK | REPLICAS | THREADS | XCF
: Display various information about the LDAP server. See "Displaying performance information and server settings" on page 184 for a description of the DISPLAY output.

LOG WRITEOPS | ALLOPS | SUMMARY | TIME | NOTIME | MERGEDRECORD | MSGS | NOMSGS | FLUSH | STOP | ROLLOVER | FILTER, *filter*
: Turn LDAP server activity logging on or off and control what server activities are logged. See "Activity logging" on page 193 for more information.

MAINTMODE ON | OFF
: Change the LDAP server between normal mode and maintenance mode. Access to the LDAP server is restricted to certain users when in maintenance mode, therefore, this mode can be used to "fix" the LDAP server. See "Basic replication maintenance mode" on page 475 or "Advanced replication maintenance mode" on page 536 for more information.

REFRESH DB2RUNSTATS | SSL

: When **DB2RUNSTATS** is specified, refresh the TDBM or GDBM (DB2-based) database statistics. This causes the LDAP server to reexamine the current database statistics recorded in the DB2 catalog by the

RUNSTATS utility. This is useful if the catalog statistics have been changed by running the **RUNSTATS** utility after the LDAP server completed initialization. See Chapter 31, "Performance tuning," on page 599 for detailed information about running the DB2 **RUNSTATS** utility.

When **SSL** is specified, initialize the SSL environment again. This might be necessary, for example, if SSL replaces expired certificates.

RESET LOCKS | MONITOR | THREADS | WLMEXCEPT *[,opid]*
If **LOCKS, MONITOR**, or **THREADS** are specified, then various counters maintained by the LDAP server are reset. See "Displaying performance information and server settings" on page 184 for the counters that are reset.

If **WLMEXCEPT** is specified without an *opid* (Operations Monitor ID), the LDAP server defaults to using only the initial **wlmExcept** configuration option settings. If an *opid* is specified, this command undoes all previous **WLMEXCEPT** operator modify commands issued for this particular *opid*. The LDAP server defaults to using the **wlmExcept** configuration options for this particular *opid* only.

SNAP Take a SNAP dump of the LDAP server. The dump is taken using the following options:

```
THR(ALL) TRACE NOFILE NOVAR NOBLOCK NOSTOR FNAME(CEEDUMP)
```

This command is only valid when the LDAP server is started in 31-bit mode.

UNLOCK ADMIN
Unlocks the LDAP root administrator entry in an LDBM, TDBM, or CDBM backend when the password has expired or the maximum number of failed bind attempts in the effective password policy has been exceeded.

After successful completion of this command, the LDAP root administrator must change their password before authenticating to the LDAP server. This command is only valid for unlocking the LDAP root administrator that is specified in the configuration file (**adminDN** configuration option) and exists as an LDBM, TDBM, or CDBM backend entry with a **userPassword** value. This command is not valid when the LDAP root administrator's entry is participating in native authentication, resides in the SDBM backend, or the password is specified in the configuration file in the **adminPW** configuration option.

WLMEXCEPT *tname,opid*
Associate a search pattern in the operations monitor entry to a WLM transaction name.

The *tname* specifies the WLM transaction name as the *opid* specifies the Operations Monitor ID value from the operations monitor entry. The *opid* indicates which search pattern is used from the operations monitor entry. This command causes any client search request matching the specified search pattern to run under the enclave with the specified WLM transaction name. Each time the **WLMEXCEPT** operator modify command is issued, the new association is added before any of the configured **wlmExcept** configuration options or previously issued **WLMEXCEPT** operator modify commands. See "Workload manager (WLM)" on page 602 for more information about running the LDAP server with WLM.

Chapter 11. Migrating to z/OS

This topic contains information about migration issues. Your plan for migrating to z/OS should include information from various sources. These sources of information describe topics such as coexistence service and optional migration actions. See the *z/OS V2R2 Migration* book on how to migrate from various z/OS releases. Also, see the *z/OS Summary of Message and Interface Changes* book 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 the z/OS elements.

- *z/OS Planning for Installation*

  This information 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 V2R2 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/etc/ds.conf file, data sets supplied, jobs, or procedures that have been completed for you, and product status. IBM might have run jobs or made updates to PARMLIB or other system control data sets. These updates might affect your migration.

Within this topic, you can find information about the specific updates and considerations that apply to this release of z/OS LDAP.

- "Migration roadmap" on page 212

  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 Version 2 Release 2 overview" on page 212

  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 might be considered, and where you can find more detailed information in the z/OS LDAP library or other element libraries.

Actions required for migrations from previous releases of z/OS

There are no required LDAP migration actions from z/OS V1R11 or z/OS V1R12 to z/OS V1R13 or z/OS V2R2.

Fallback from a TDBM or DB2-based GDBM backend in z/OS IBM TDS to an earlier z/OS IBM TDS version

You might find it necessary to return to an earlier z/OS IBM TDS version after migrating to the current version of z/OS IBM TDS. Follow these procedures before running on a previous version of the server that is using the TDBM or DB2-based GDBM backends.

If you need to fallback to z/OS V1R11 IBM TDS, the fallback procedure is as follows if the **serverCompatLevel** configuration option is not set to **5** in the server configuration file. (If the **serverCompatLevel** configuration option is already set to **5**, the server is already ready to run on z/OS V1R11 and it is not necessary to follow this fallback procedure):

1. Stop z/OS IBM TDS, if it is running.
2. Set the **serverCompatLevel** configuration option to **5**.
3. Restart z/OS IBM TDS on the current z/OS release and then shut it down. It is now ready to run on z/OS V1R11 with the TDBM and DB2-based GDBM backends.

If you need to fallback to z/OS V1R12 or V1R13 IBM TDS, the fallback procedure is as follows if the **serverCompatLevel** configuration option is not set to **6** in the server configuration file. (If the **serverCompatLevel** configuration option is already set to **6**, the server is already ready to run on z/OS V1R12 and it is not necessary follow this fallback procedure):

1. Stop z/OS IBM TDS, if it is running.
2. Set the **serverCompatLevel** configuration option to **6**.
3. Restart z/OS IBM TDS on the current z/OS release and then shut it down. It is now ready to run on z/OS V1R12 with the TDBM and DB2-based GDBM backends.

LDAP_COMPAT_FLAGS environment variable

The **LDAP_COMPAT_FLAGS** environment variable provides a server compatibility knob that can be set temporarily while migrating from the z/OS Integrated Security Services LDAP server to IBM TDS for z/OS or from an earlier IBM TDS for z/OS release to the current release. This environment variable provides some limited support for restoring behavior that existed in the z/OS Integrated Security Services LDAP server or an earlier IBM TDS for z/OS release. This environment variable is only meant for temporary use when LDAP client applications rely on data being returned in a certain format. Ultimately these LDAP client applications must be updated to support IBM TDS for z/OS behavior.

The **LDAP_COMPAT_FLAGS** level is a mask that you can specify in the following ways:

- A decimal value (for example, `1`)
- A hexadecimal value (for example, `x02` or `X02`)
- A keyword (for example, `SDBM_MIXEDDN`)
- A construct of these values by using plus and minus signs to indicate inclusion or exclusion of a value. For example:
  - '1+2' is the same as specifying 'x01+x02', or 'SDBM_MIXEDDN+SDBM_ISS_SEARCH'

– '2147483647-16' is the same as specifying 'x7FFFFFFF-x00000001' or 'ANY-SDBM_MIXEDDN'

Table 19 includes the **LDAP_COMPAT_FLAGS** levels and the related decimal, hexadecimal, and keyword values.

Table 19. **LDAP_COMPAT_FLAGS** *levels and support*

Keyword	Decimal level	Hexadecimal level	Description
OFF	0	0x00000000	No compatibility level setting

SDBM_MIXEDDN	1	0x00000001	On SDBM backend search responses, attribute values that are DN-type attribute values are returned in mixed case format, which is how the z/OS Integrated Security Services LDAP server returned these attribute values. By default, IBM TDS for z/OS returns these attribute values in uppercase format.
			
			The SDBM attribute types and values affected by this setting are: **racfNotify, racfGroupUserids, racfConnectGroupName, racfDefaultGroup, racfSubGroupName, racfSuperiorGroup, racfConnectOwner, racfOwner, racfStdataUser, racfStdataGroup, racfAudit, racfCdtInfoGroup, racfCdtInfoMember, racfClassAct, racfGenCmd, racfGeneric, racfGenList, racfGlobal, racfLogoptionsAlways, racfLogoptionsDefault, racfLogoptionsFailures, racfLogoptionsNever, racfLogoptionsSuccesses, racfRacList,** and **racfStatistics**
			
			Mixed case format examples with the SDBM suffix set to `cn=sdbm`:
			`racfid=X,profiletype=USER,cn=sdbm`
			`racfid=Y,profiletype=GROUP,cn=sdbm`
			`racfuserid=X+racfgroupid=Y,`
			`profiletype=CONNECT,cn=sdbm`
			`profilename=TMP,profiletype=FACILITY,cn=sdbm`
			
			Uppercase (default) format examples with the SDBM suffix set to `cn=sdbm`:
			`RACFID=X,PROFILETYPE=USER,CN=SDBM`
			`RACFID=Y,PROFILETYPE=GROUP,CN=SDBM`
			`RACFUSERID=X+RACFGROUPID=Y,`
			`PRFILETYPE=CONNECT,CN=SDBM`
			`PROFILENAME=TMP,PROFILETYPE=FACILITY,CN=SDBM`

SDBM_ISS_SEARCH	2	0x00000002	For SDBM base or subtree level searches where the base DN is an SDBM user that does not exist, the LDAP server sets the LDAP return code to **LDAP_OTHER** and reason code message to ICH30001I. This is for compatibility purposes with the z/OS Integrated Security Services LDAP server. By default in this situation, IBM TDS for z/OS sets the LDAP return code to **LDAP_NO_SUCH_OBJECT**, a reason code of R004071, and a matchedDN set to "profiletype=user,`SDBM_suffix`". Also, this setting affects the format of the returned **objectclass** attribute values for the SDBM suffix entry, entries under and including the `profiletype=user,SDBM_suffix`, `profiletype=group,SDBM_suffix`, and `profiletype=connect,SDBM_suffix` to be in mixed case format. This is for compatibility purposes with the z/OS Integrated Security Services LDAP server. By default, IBM TDS for z/OS returns these **objectclass** attribute values in uppercase format.
ANY	2147483647	0xFFFFFFFF	Turns on all compatibility settings.
ALL	2147483647	0xFFFFFFFF	Turns on all compatibility settings.

Updating LDAP configurations settings in a sysplex without server outage

This section describes updating configuration option settings for the LDAP servers that run in a sysplex without server outage. For example, you can update the **serverCompatLevel** configuration option to change supported functions.

In general, for those options that are not sysplex-related, you can change them and restart this specific server for changes to take effect, and the sysplex service is not impacted. However, you might want some sysplex level configuration changes such as:

- Upgrade the server compatibility level.
- Revert to an earlier version of the server compatibility level.
- Add new backends.
- Remove existing backends. This does not include SDBM, which is not sysplex-related

To ensure that the entire sysplex is able to handle LDAP requests during the update, a new server needs to be started in transition mode. For more information about how to start a server in transition mode, see "Setting up and running the LDAP server" on page 171. For the convenience of description, the server that is running in transition mode is called a transition server. The entire task when you update the configuration options setting in a sysplex is called transition work. The transition server that is running in transition mode is in a special intermediate state to handle the LDAP requests while other LDAP servers in the sysplex group are having updates done to their configurations. The transition server is restored to normal work mode after the entire sysplex update completes.

The steps to take are:

1. Back up your database files.
2. Prepare the new configuration file that contains the new setting of the options you want to update.
3. Start a new LDAP server in transition mode as transition server with the updated configuration file. You can also shut down an existing LDAP server of the sysplex and restart it as transition server. For transition server, the **serverCompatLevel** configuration option must be set even if you want to upgrade the backends with the same compatibility level as before. The current setting can be found in the startup messages of the sysplex owner.

   The transition server works with the previous **serverCompatLevel** configuration option and backend configurations before it becomes owner.
4. Shut down the non-owner server first, and then the current sysplex owner server to avoid unnecessary owner switching. The transition server then becomes owner and begins processing requests by using the new **serverCompatLevel** and backend configurations.
5. Start other servers with updated configurations.
6. Shut down the transition server, or restart it in normal (non-transitional) mode and transition completes.

Notes:

1. Select the time with the least number of client requests that are handled by the LDAP server to do sysplex level updates.
2. The prerequisite to do transition work is that all servers in the target sysplex group are to be run with z/OS Version 2 Release 2 or higher, including the transition server itself and compatibility level of 4 or higher. If not, you must upgrade to the appropriate z/OS release and compatibility level first.

3. If you want to undo the transition or lower the server compatibility level, any DIT-related updates that result from the compatibility level upgrade must be undone manually.
4. If the transition involves removing backends from the configuration, any LDAP request for the eliminated backends fails if it is directed into the transition server before the transition server becomes the owner. Also, it fails no matter which server it is directed to when the transition completes. If any error occurs during the transition, the transition server terminates, and is an LDAP service outage.
5. If any error occurs during the transition, the transition server ends and an LDAP service outage occurs.
6. ARM is not enabled after the transition completes, even though it was enabled before the transition.
7. Not all configuration options support sysplex level updates without outages. The transition server is not able to join the current sysplex group when the following options are updated:
   - **schemaPath**
   - **serverName**
   - **serverSysplexGroup**

   Notes:
   - The **schemaPath** must be set with the same value within the whole sysplex.
   - The **serverName** must be the same for all TDBM and DB2-based GDBM backends.
8. Because changing the suffixes (except TDBM), the database path or **dbuserid** for the existing backends are not supported for the transition. The transition server fails to start if any change is detected for an existing backend name. If you want to remove one or more backends and add new backends, do not reuse the backend names of any that are removed.
9. If the transition involves removing backends from the configuration, make sure that there are no replication topologies that are related to the backends that are removed. Otherwise, delete the corresponding replication topologies first.

   If you remove replication topology entries under the **cn=localhost** suffix such that this suffix is no longer needed, and the suffix is in an LDBM backend that is not being removed, you cannot remove the suffix from the configuration in transition mode. To remove a suffix from an LDBM backend, all servers in the target sysplex group must be taken down simultaneously.

Checking file ownership for the LDAP server

APAR OA43909 introduces checking of file ownership for the LDAP server and includes the **ds2ldif** and **ldif2ds** utilities. LDAP backend directories and their files are checked before the files are opened.

After this APAR is applied, make sure that:

- The z/OS UNIX System Services directories and files are owned by the user ID that runs the LDAP server or the user ID must be in the group that owns the directory and its files.
- Before the LDAP server is restarted, ensure that the user ID that runs the LDAP server owns the directories and their files.
- Before the **ds2ldif** or **ldif2ds** utilities are run, ensure that the user ID that runs the utilities is in the group that owns directories and their files.

For more information about file ownership, see:

- "Additional setup for user ID that runs the LDAP server" on page 48

- "Setting up a user ID for your LDAP server" on page 45
- "ds2ldif utility" on page 225
- "ldif2ds utility" on page 235

Migration roadmap

This section describes the migration paths that are supported by the current release of z/OS .

z/OS Version 2 Release 2 update summary

The following table summarizes the updates introduced in z/OS Version 2 Release 2. If you are migrating from releases before z/OS Version 2 Release 2, review the information in the detailed section for each item.

z/OS Version 2 Release 2 overview

This section describes the new and changed server functions introduced for z/OS Version 2 Release 2. The information about each item includes:

- Description
- Summary of the z/OS server tasks or interfaces that might 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.

Table 20. Summary of IBM TDS updates for z/OS Version 2 Release 2

For information about:	See topic:
Compatibility level upgrade without an LDAP outage	"Updating LDAP configurations settings in a sysplex without server outage" on page 210
Activity logging	"Activity logging" on page 193
Dynamic group performance and scalability	"User groups considerations in large directories" on page 621
Replication of password policy attributes	"Replicating password policy operational attributes" on page 383

Compatibility level upgrade without an LDAP outage:

Description: This support allows improved availability when migrating new function across a sysplex. To update the **serverCompatLevel** or backends in a sysplex group without LDAP service outage, a run mode (transition mode) is now added. The LDAP instance can now be started in transition mode and continue to be operational while others are updated and recycled. In transition mode, the server can be started successfully with the updated **serverCompatLevel** or backends configurations. Transition mode is a temporary work mode that is used during sysplex level configuration updates.

What this change affects: Migration of new function across LDAP servers in a sysplex can be done with improved availability. A rolling migration to a new server compatibility level can be accomplished without a complete outage of all LDAP servers in the sysplex group.

Dependencies: Migration activity using transition mode requires that all servers in the sysplex group that are active during the transition be running on z/OS Version 2 Release 2 or later.

Coexistence considerations: None.

Migration tasks: None.

For more information: See "Updating LDAP configurations settings in a sysplex without server outage" on page 210 for more information.

Activity logging:

Description: z/OS Version 2 Release 2 enhancements to activity logging are:

- Additional client events are now logged. This includes connect, disconnect, abandon requests, and unknown requests.
- Additional data is provided in existing log records. This includes attribute names in add requests, attribute names, and modification operation type in modify requests, and message ID in all request records.
- Improved consistency between the data that is captured in activity logging and SMF auditing.
- Versioning of the activity log record contents to avoid compatibility and migration issues when functional enhancements change the data contents of the activity log.
- Additional configuration options, eliminating the need to use environment variables to control some of the configuration options.

What this change affects: Additional details are available in activity logging related to client events.

Dependencies: None.

Coexistence considerations: None.

Migration tasks: None.

For more information: See "Activity logging" on page 193 for more information.

Dynamic group performance and scalability:

Description: The support for dynamic groups is redesigned to improve performance when determining a user's groups. Scalability is improved, allowing administrators to define a large number of dynamic groups without experiencing performance issues.

This improvement is automatically enabled. No configuration task is necessary.

What this change affects: Performance is improved when gathering groups after a bind operation or when retrieving the **ibm-allGroups** operational attribute on a search or compare operation.

Dependencies: None.

Coexistence considerations: None.

Migration tasks: None.

For more information: See "User groups considerations in large directories" on page 621 for more information.

Replication of password policy attributes:

Description: Read-only replica password policy replication support provides consistent replication updates of password policy operational attributes on all servers when they are updated in a read-only server in a replication topology. As a result, this new feature reduces the opportunity for circumvention of password policies while it ensures functional equivalence across Tivoli Directory Server distributed and z/OS. A new attribute, **ibm-slapdReplicateSecurityAttributes**, is added to `cn=replication, cn=configuration` for it to enable this feature.

What this change affects: Password policy operational attributes in user entries are maintained more consistently across replicas when using advanced replication.

Dependencies: A server compatibility level of 8 or higher is required to replicate password policy operational attribute updates that occur on read-only replicas during authentication.

Coexistence considerations: None.

Migration tasks: None.

For more information: See "Replicating password policy operational attributes" on page 383 for more information.

z/OS Version 2 Release 1 update summary

The following table summarizes the updates introduced in z/OS V2R1. If you are migrating from releases before z/OS V2R1, review the information in the detailed section for each item.

z/OS Version 2 Release 1 overview

This section describes the new and changed server functions introduced for z/OS V2R1. The information about each item includes:

- Description
- Summary of the z/OS server tasks or interfaces that might 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.

Table 21. Summary of IBM TDS updates for z/OS V2R1

For information about:	See topic:
Remote crypto plug-in	Remote crypto plug-in
ICTX plug-in	ICTX plug-in
TLS (Transport Layer Security) V1.2 support	• "Enabling SSL/TLS support" on page 70 • *z/OS Cryptographic Services System SSL Programming*

Remote crypto plug-in:

Description: The z/OS LDAP server provides client applications that need access to a PKCS #11 implementation with the support provided in the remote crypto plug-in. PKCS #11 is one of the cryptographic standards of Public Key Cryptographic Standards (PKCS) that define a platform independent API to cryptographic tokens. The PKCS #11 standard defines the types of cryptographic

tokens and how to use, create, and delete tokens, including how to encrypt, decrypt, and hash data with those tokens.

What this change affects: PKCS #11 functions can now be performed by the z/OS LDAP server with ICSF.

Dependencies: ICSF must be installed and running.

Coexistence considerations: None.

Migration tasks: None.

For more information: See for more information. Also, see:

- *z/OS Cryptographic Services ICSF Application Programmer's Guide*
- *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications*

ICTX plug-in:

Description: The ICTX plug-in provides centralized remote resource management. This allows resource managers that do not exist on z/OS to centralize authorization decisions and security event logging by using RACF through the ICTX plug-in.

What this change affects: Allows remote authorization and auditing to occur through RACF.

Dependencies: None.

Coexistence considerations: None.

Migration tasks: None.

For more information: See ICTX plug-in for more information.

TLS (Transport Layer Security) V1.2 support:

Description: Support is provided for TLS (Transport Layer Security) V1.1 protocol, TLS V1.2 protocol, and NSA Suite B profile.

What this change affects: Allows z/OS IBM® Tivoli® Directory Server, its LDAP client, and its command-line utilities to use TLS V1.1 and TLS V1.2 protocols, and NSA Suite B profile for secure connections using z/OS System SSL.

Dependencies: None.

Coexistence considerations: None.

Migration tasks: None.

For more information: See "Enabling SSL/TLS support" on page 70 and *z/OS Cryptographic Services System SSL Programming*.

Chapter 12. Running and using the LDAP server utilities

Utility programs are provided to assist in initializing, backing up, and synchronizing the data managed by the LDAP server.

Operation	Server utility
Encrypt passwords in a backend directory	**db2pwden**
Unload data from backend directory to an LDIF file	**ds2ldif**
Load data into backend directory	**ldif2ds (TDBM only)**
Identify differences between two directory subtrees on two different servers and optionally synchronize the servers	**ldapdiff**
Perform extended operations	**ldapexop**

The **db2pwden, ds2ldif, ldif2ds**, and **ldapexop** utilities can be run in the z/OS shell, as jobs using JCL and procedures, or from TSO. The **ldapdiff** utility can only be run in the z/OS shell.

Format and usage information for the utilities are in:
- "db2pwden utility" on page 222
- "ds2ldif utility" on page 225
- "ldif2ds utility" on page 235
- "ldapdiff utility" on page 247
- "ldapexop utility" on page 255

See "Creating the DB2 database and table spaces for TDBM or GDBM" on page 55 for information about the permissions necessary to run the **ds2ldif** and **ldif2ds** utilities with a TDBM backend.

Running the LDAP server utilities in the z/OS shell

The **ds2ldif, ldapdiff**, and **ldif2ds** server utilities can be run from the shell using sample script files shipped in **/usr/lpp/ldap/sbin**. The **db2pwden** and **ldapexop** utilities are shipped as executable files in **/usr/lpp/ldap/sbin**.

See Table 22 for sample script file names and descriptions:

Table 22. Sample script file

z/OS script file names	Description
ds2ldif31	Used to unload data from an LDBM, TDBM, CDBM, or schema backend in 31-bit mode
ds2ldif64	Used to unload data from an LDBM, TDBM, CDBM, or schema backend in 64-bit mode
ldapdiff	Used to compare data between a master and replica server and to synchronize replica servers
ldif2ds or **ldif2ds31**	Used to bulk load LDIF data into a TDBM backend in 31-bit mode
ldif2ds64	Used to bulk load LDIF data into a TDBM backend in 64-bit mode

Before running these server utilities, modify the sample script files in Table 22 on page 217 for your environment. You should set **STEPLIB** to SYS1.SIEALNKE, if the PDS has not been placed in LNKLST. If you are loading or unloading entries in a DB2-based backend and your runtime libraries for DB2 are not in LNKLST or LPA on the system, you must specify the DB2 high-level qualifier for your DB2 installation in the exported **STEPLIB** in the **ldif2ds, ldif2ds31, ldif2ds64, ds2ldif31**, and **ds2ldif64** script files. These script files require access to the *DB2HLQ*.SDSNLOAD data set. In addition, access to the *DB2HLQ*.SDSNLOD2 data set is required for **ldif2ds64** and **ds2ldif64**.

After the sample script files are updated, update the **PATH** environment variable to specify **/usr/lpp/ldap/sbin:/usr/sbin** to enable easy access to the script files and server utility executables shipped in the LDAP server.

If you are using the **ldif2ds** utility to add an entry containing **userPassword, secretKey, ibm-replicaKeyPwd, replicaCredentials, ibm-slapdMasterPw**, or **ibm-slapdAdminPw** attribute values that are encrypted or hashed using ICSF, the **LIBPATH** variable might need to be set. See "Installing ICSF for encryption, hashing, or SSL/TLS" on page 22 for more information.

If you are using the **ds2ldif** or **ldif2ds** utilities and the **dsnaoini** option is not specified in the LDAP server configuration file, the **DSNAOINI** environment variable might need to be exported before running these utilities. If the **DSNAOINI DD** card is specified in the LDAP server procedure, set the **DSNAOINI** environment variable to the same data set before using the **ds2ldif** or **ldif2ds** utilities.

When started, **db2pwden, ds2ldif31, ds2ldif64, ldif2ds, ldif2ds31**, and **ldif2ds64** read an environment variables file. The default file is **/etc/ldap/ds.envvars**. This default can be changed by setting the environment variable, **LDAP_DS_ENVVARS_FILE**, to the fully qualified z/OS UNIX System Services path name, the sequential data set, or the partitioned data set. Some of the environment variables that can be set are **NLSPATH** and **LANG**.

Running the LDAP server utilities from JCL

Sample JCL for running **db2pwden, ds2ldif, ldif2ds**, and **ldapexop** from batch is provided with the LDAP server. Installations must modify the included JCL inline procedure to ensure that the appropriate load modules can be found. It might 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 data set and entering the **submit** command. The member names are **DB2PWDEN** (for **db2pwden**), **DS2LDF31** (for using **ds2ldif** in a 31-bit server environment), **DS2LDF64** (for using **ds2ldif** in a 64-bit mode), **LDF2DS** or **LDF2DS31** (for using **ldif2ds** in 31-bit mode), **LDF2DS64** (for using **ldif2ds** in 64-bit mode), and **LDAPEXOP** (for **ldapexop**).

If you are loading or unloading entries in a DB2-based backend and your runtime libraries for DB2 are not in LNKLST or LPA on the system, you must specify the DB2 high-level qualifier for your DB2 installation in a **STEPLIB DD** card in the **LDF2DS, LDF2DS31, LDF2DS64, DS2LDF31**, or **DS2LDF64** batch job. These batch jobs require access to the *DB2HLQ*.SDSNLOAD data set. In addition, access to the *DB2HLQ*.SDSNLOD2 data set is required for **LDF2DS64** and **DS2LDF64**.

Running the LDAP server utilities in TSO

The **db2pwden, ds2ldif, ldapexop**, and **ldif2ds** server utilities can be run from TSO. In order to do this, some elements of the environment must be set up to locate the LDAP programs. Following are the steps to do this:

1. The PDS (SYS1.SIEALNKE) where the LDAP server load modules were installed must be accessible through **LNKLIB, LPALIB**, or specified on the **TSOLIB** command.

   ```
   tsolib act dsn('SYS1.SIEALNKE')
   ```

2. The PDS (*GLDHLQ*.SGLDEXEC) containing the CLISTs needed to run the utilities must be available in SYSEXEC:

   ```
   alloc f(SYSEXEC) da('GLDHLQ.SGLDEXEC')
   ```

3. The default environment variables file for the utilities can be changed by creating a data set to hold the environment variables and then using the TSO **alloc** command as shown:

   ```
   alloc f(ENVVAR) da('datasetname')
   ```

If you want to specify an LDAP server configuration file that is a data set, you can specify the **-f** option on the command. For example:

```
ds2ldf31 -f "//'datasetname'" -o /tmp/ldif.1
```

Alternately, to specify the LDAP server configuration file by associating it with a DD name, enter in TSO:

```
alloc da('datasetname') fi(config) shr
```

and then start the utility without specifying the **-f** option.

After this setup is complete, running these utilities follows the same syntax as would be used if running them in z/OS, except that the program names are eight characters or less. See "Running the LDAP server utilities in the z/OS shell" on page 217 for more information. To run these utilities from TSO, use the following names:

Table 23. Names for running LDAP server utilities from TSO

z/OS shell name	TSO name
db2pwden	db2pwden
ds2ldif31	ds2ldf31
ds2ldif64	ds2ldf64
ldapexop	ldapexop
ldif2ds	ldf2ds
ldif2ds31	ldf2ds31
ldif2ds64	ldf2ds64

SSL/TLS information for LDAP utilities

The contents of a client's key database file is managed with the **gskkyman** utility. See *z/OS Cryptographic Services System SSL Programming* for information about the **gskkyman** utility. The **gskkyman** utility is used to define the set of trusted certificate 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_sasl_bind_s()** API.

For example, if the LDAP server is using a high-assurance VeriSign certificate, 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 **db2pwden** or **ldapexop** utilities without the **-Z** parameter or the **ldapdiff** utility without the **-sZ** or the **-cZ** parameters and calling the secure port on an LDAP server (in other words, a nonsecure call to a secure port), is not supported. Also, a secure call to a nonsecure port is not supported.

SSL/TLS encrypts the key database file, therefore, either the key database password or a stash file must be specified on the **-P** *keyFilePW* parameter. If a stash file is specified, it must be specified in the form `file://` followed immediately (no blanks in between) by the file specification of the stash file. See *z/OS Cryptographic Services System SSL Programming* for information about using the **gskkyman** utility to create a 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 Cryptographic Services System SSL 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
```

After the RACF key ring is set up and authorized, specify the RACF key ring name for the **-K** *keyFile*, **-cK** *keystore*, or **-sK** *keystore* options and do not specify the **-P** *keyFilePw*, **-cP** *keystorePwd*, or **-sP** *keystorePwd* options.

Using PKCS #11 tokens

The **db2pwden** and **ldapexop** utilities support the use of PKCS #11 tokens. The **gskkyman** utility or the **RACDCERT** command can be used to create or modify PKCS #11 tokens. ICSF uses the CRYPTOZ SAF class to determine if the issuer of

the **gskkyman** utility or the **RACDCERT** command is permitted to perform the operation against a z/OS PKCS #11 token. See *z/OS Cryptographic Services System SSL Programming* for more information about the **gskkyman** utility and *z/OS Security Server RACF Command Language Reference* for more information about the **RACDCERT** command.

The user ID that runs the **db2pwden** or **ldapexop** utility must be authorized by RACF to use the PKCS #11 token. To authorize the user ID, you can use the RACF commands in the following example (where *NAME* is the name of the PKCS #11 token and *userid* is the user ID running the utility).

```
SETROPTS CLASSACT(CRYPTOZ)
RDEFINE CRYPTOZ USER.NAME UACC(NONE)
RDEFINE CRYPTOZ SO.NAME UACC(NONE)
PERMIT USER.NAME CLASS(CRYPTOZ) ID(userid) ACCESS(READ)
PERMIT SO.NAME CLASS(CRYPTOZ) ID(userid) ACCESS(READ)
```

Remember to refresh RACF after doing the authorizations.

```
SETROPTS RACLIST(CRYPTOZ) REFRESH
```

Once the PKCS #11 token is set up and authorized, specify the PKCS #11 token for the **-K** *keyFile* option in the following manner: `-K *TOKEN*/NAME`. Also, do not specify the **-P** *keyFilePW* option when using a PKCS #11 token.

Using a Java keystore or RACF key ring for ldapdiff

Keys and certificates needed for using SSL with the **ldapdiff** utility must be stored in either a Java™ keystore file (.jks) or a RACF key ring. If your existing keys and certificates are in a System SSL key database file or a PKCS #11 token, you must migrate them to a keystore file or a RACF key ring. If they are already in a RACF key ring, you can use them as is or can migrate them to a keystore file. To migrate keys and certificates, you must first export them from their source location and then import them into their target location.

You can use the **gskkyman** utility to export from a key database file or from a PKCS #11 token. You can use the **RACDCERT** command to export from a RACF key ring or PKCS #11 token.

You can then use the **RACDCERT** command to import into a RACF key ring or the **keytool** Java application to import into a Java keystore. Make sure to specify the `-storetype JCEKS` option when using **keytool**.

When set up is complete, identify the Java keystore or RACF key ring using the appropriate **-sK** *keystore*, **-sN** *keystoreType*, **-cK** *keystore*, **-cN** *keystoreType*, **-sT** *truststore*, **-st** *truststoreType*, **-cT** *truststore*, or **-ct** *truststoreType* options when running the **ldapdiff** utility.

See *z/OS Cryptographic Services System SSL Programming* for more information about the **gskkyman** utility and migrating keys and certificates. See *z/OS Security Server RACF Command Language Reference* for more information about the **RACDCERT** command. See the *keyTool User Guide for SDK* for more information about the **keytool** Java application.

Utility programs

This section describes the **db2pwden, ds2ldif, ldif2ds, ldapdiff**, and **ldapexop** server utilities.

db2pwden utility

Purpose

The **db2pwden** utility is provided to encrypt or hash all unencrypted, AES encrypted, and DES encrypted user passwords in an already loaded TDBM, LDBM, or CDBM backend. The utility runs as a client operation while the server is active, and causes the server to encrypt or hash all the **userPassword** attribute values that are unencrypted, AES encrypted, or DES encrypted with the **pwEncryption** method configured on the LDAP server. The utility must be run by an LDAP administrator with the appropriate authority or a user with the authority to update password values. See "Administrative roles" on page 159 for more information about administrative role authority.

Format

db2pwden [*options***]**

Parameters

options

The following table shows the options that you can use for the **db2pwden** utility:

Table 24. db2pwden options

Option	Description
-?	Print this text.
-b *baseDN*	Use *baseDN* 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 *baseDN* definition. If you are running in TSO, set the **LDAP_BASEDN** environment variable using Language Environment® runtime environment variable **_CEE_ENVFILE**. See *z/OS XL C/C++ Programming Guide* for more information. If you are running in the z/OS shell, export the **LDAP_BASEDN** environment variable.
-d *debugLevel*	Specify the level of debug messages to be created. The debugLevel is specified in the same fashion as the debug level for the LDAP server. LDAP debug levels lists the specific debug levels. The default is no debug messages.
-D *bindDN*	Use *bindDN* to bind to the LDAP directory. The *bindDN* parameter is 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 is used for making access checks. This directive is optional when used in this manner.
-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.
-h *ldapHost*	Specify the host name or IP address on which the LDAP server is running. The default is the local host.

Table 24. db2pwden options (continued)

Option	Description
-K *keyFile*	Specify the name of the System SSL key database file, RACF key ring, or PKCS #11 token. If this option is not specified, this utility looks for the presence of the `SSL_key ring` environment variable with an associated name. If *keyFile* is specified as `*TOKEN*/NAME`, then System SSL uses the specified PKCS #11 token. Otherwise, System SSL uses a key database file or a RACF key ring. In this case, System SSL first assumes that *keyFile* is a key database file name and tries to locate the file. If *keyFile* is not a fully qualified z/OS UNIX System Services file name, the current directory is assumed to contain the key database file. The name cannot be a data set. If System SSL cannot locate the file, it then assumes that *keyFile* is a RACF key ring name. See "SSL/TLS information for LDAP utilities" on page 219 for information about System SSL key databases, RACF key rings, and PKCS #11 tokens. This parameter is ignored if **-Z** is not specified.
-m *mechanism* or **-S** *mechanism*	Specify the bind method to use. You can use either **-m** or **-S** to indicate the bind method. Specify `GSSAPI` to indicate that a Kerberos Version 5 bind 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 that a SASL digest 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. Unless you want to use the default certificate in the key database file, RACF key ring, or PKCS #11 token, use the **-N** option to specify the label of the certificate. 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. If **-m** or **-S** is specified, a simple bind is performed.
-N *keyFileDN*	Specify the label associated with the certificate in the key database file, RACF key ring, or PKCS #11 token. This parameter is ignored if **-Z** is not specified
-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.
-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 file system file specification (for example, file:///etc/ldap/sslstashfile). The stash file must be a file and cannot be a data set. This parameter is ignored if **-Z** is not specified.

Table 24. db2pwden options (continued)

Option	Description
-U *userName*	Specify the *userName* for `CRAM-MD5` or `DIGEST-MD5` binds. The *userName* is a short name (uid) that is used to perform bind authentication. This option is required if the **-S** or **-m** option is set to `DIGEST-MD5`.
-w *passwd*	Use *passwd* as the password for simple, `CRAM-MD5`, and `DIGEST-MD5` authentication. The default is a NULL string.
-v	Use verbose mode, with many diagnostics written to standard output.
-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 a file system key database file. Unless you want to use the default certificate in the key database file, RACF key ring, or PKCS #11 token, use the **-N** option to specify the label of the certificate.

All other command-line inputs result in a syntax error message and the correct syntax is displayed. If the same option is specified multiple times or if both **-m** and **-S** are specified, the last value specified is used.

The **db2pwden** utility sends the **PasswordPolicy** control as a non-critical control when the user attempts to authenticate to the targeted LDAP server. If the bound user is subject to password policy on the LDAP server, the **db2pwden** utility parses and displays the warning and error messages from the **PasswordPolicy** control response.

Examples

Following are examples using the **db2pwden** utility:

- The following command:

  ```
  db2pwden -D "cn=admin" -w secret
  ```

  encrypts or hashes all unencrypted, AES encrypted, or DES encrypted passwords in the TDBM, LDBM, or CDBM backend at the LDAP server on the local host. The base is defined in the **LDAP_BASEDN** environment variable. The encryption or hashing 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 or hashes all unencrypted, AES encrypted, or DES encrypted passwords starting at the base `"o=university,c=US"` in the TDBM or LDBM backend on host `ushost` at port `391`. The encryption or hashing method used is the **pwEncryption** method configured on the LDAP server.

Diagnosis

Exit status is 0 if no errors occur. Errors result in a nonzero exit status and a diagnostic message being written to standard error.

ds2ldif utility

Purpose

The **ds2ldif** utility is used to unload entries from a directory that is stored in a TDBM, LDBM, or CDBM backend into a file in LDAP Data Interchange Format (LDIF). The utility is also used to obtain the LDAP server schema entry. **ds2ldif** cannot be used to unload a GDBM or SDBM backend.

There are two versions of the **ds2ldif** utility:
- **ds2ldif31** (runs in a 31-bit environment)
- **ds2ldif64** (runs in a 64-bit environment)

Both can be used to unload LDBM entries, TDBM entries, CDBM entries, or the LDAP server schema entry.

If using the **ds2ldif** utility in an interoperability environment, see Appendix G, "Guidelines for interoperability between non-z/OS TDS and z/OS TDS," on page 743 for more information.

Format

```
ds2ldif31 | ds2ldif64 -o outputFile [-d debugLevel] [-f confFile] [-g]
                       [-j] [-k keyFile] [-r [-q filterDN]]
                       [[-s subtreeDN] | [-n beName] [-l]] [-t]
                       [-w adminPW] [-Z] [-?]
```

Parameters

The following table shows the options that you can use for the **ds2ldif** utility:

Option	Description
-?	Display the usage.
-d *debugLevel*	Specify the level of the debug messages to be created. The debugLevel is specified in the same fashion as the debug level for the LDAP server. LDAP debug levels lists the specific debug levels. The default is no debug messages.
-f *confFile*	Specify the name of the LDAP server configuration file to use. This configuration file only needs to contain information for the backend to be unloaded. The configuration data set specified by the CONFIG DD statement is used if the **-f** parameter is not specified. The `/etc/ldap/ds.conf` configuration file is used if the **-f** parameter is not specified and the CONFIG DD statement is not defined.
-g	Specify to unload entries in genealogical order. This unloads entries in each subtree together, doing a depth-first traversal of the directory. Specify this option when you are unloading many entries that you are loaded later, using the **ldif2ds** utility with the **-g** option, because this order of entries improves the capacity of **ldif2ds** to process large numbers of entries. Unloading in this order requires more processing and impacts unload performance.
-j	Indicate that the **replicateOperationalAttributes** control value is not written to the output LDIF file. The **replicateOperationalAttributes** control value has the **modifyTimestamp, createTimestamp, creatorsName**, and **modifiersName** attribute types and values for the entry base64 encoded.
-k *keyFile*	Specify the name of the file containing the LDAP server encryption keys. If running **ds2ldif** in batch, the data set specified by the **LDAPKEYS DD** statement is used if the **-k** option is not specified. The key file must be specified if any entries unloaded have **userPassword, secretKey, replicaCredentials, ibm-replicaKeyPwd, ibm-slapdAdmimPw**, or **ibm-slapdMasterPw** attribute values that are encrypted using the DES or AES algorithm and not using ICSF. These attribute values are decrypted and base64 encoded as the entry is written to the output LDIF file.

Option	Description
-l	Specify that entries under the **cn=localhost** suffix are unloaded from the LDBM or TDBM backend. When an entire LDBM or TDBM backend is unloaded, the entries under **cn=localhost** are not unloaded unless the **-l** option is specified. The **-l** option cannot be used when the **-s** option is specified.
-n *beName*	Specify the name of the LDBM, TDBM, or CDBM backend to unload. This is the name that is assigned to the backend on its **database** record in the LDAP server configuration file or the name that is automatically generated when the LDAP server is started. This can be used to indicate which LDBM, TDBM, or CDBM backend to process when there are multiple LDBM, TDBM, or CDBM backends in the configuration file. The **-n** option cannot be used when the **-s** option is specified.
-o *outputFile*	Specify the name of an output file to contain the unloaded directory entries. See "Specifying a value for filename" on page 86 for information about how to specify the file name.
-q *filterDN*	Specify a distinguished name (DN) of a replication filter entry which contains **ibm-replicationFilterAttr** attribute values. These values are filters that are used to skip entire entries or attributes within entries while unloading the directory. See "Partial replication" on page 536 for more information about replication filter entries. The targeted LDAP server must be running and the **-r** option must be specified when the **-q** option is specified. Also, the CDBM backend must be configured and **useAdvancedReplication on** specified in the CDBM backend section of the server configuration file to perform unload filtering.
-r	Perform an **unloadRequest** extended operation (1.3.18.0.2.12.62) to unload the subtree or backend. If the LDAP server that contains the backend that is to be unloaded is running, an **unloadRequest** extended operation can be sent to the LDAP server to unload the entries.
-s *subtreeDN*	Identify the DN of the top entry of the subtree whose entries are to be unloaded. This entry, plus all below it in the directory hierarchy, are written to the output file. The **-s** option must be used to unload the LDAP server schema entry, `cn=schema` . The **-s** option cannot be used when the **-n** option is specified.
-t	Specify that hashed **userPassword** and **ibm-slapdAdminPw** attribute values are unloaded with their encryption tag in clear text. See **Using the -t (tagging) option** for more information about this option.
-w *adminPW*	When using the **unloadRequest** extended operation, specify the password of the LDAP root administrator defined in the configuration file. Do not specify the **-w** option if the **adminPW** option is specified in the LDAP server configuration file. In this case, the value from the server configuration file is used to perform the LDAP bind before sending the **unloadRequest** extended operation to the LDAP server. If the **adminPW** configuration option is not present and the **-w** option is not specified or a **?** is specified on the **-w** option, a prompt is displayed for the LDAP root administrator password.
-Z	When using the **unloadRequest** extended operation, use SSL to encrypt the communication between the **ds2ldif** utility and the LDAP server. By default, the **ds2ldif** utility attempts to use SSL to communicate with the LDAP server assuming that the LDAP server configuration file has the necessary SSL options (for example, **sslKeyRingFile, sslKeyRingFilePW, sslCertficiate, sslKeyRingStashFile**) specified along with a secure **listen** option (for example, `listen ldaps://`). If SSL cannot be used, the **ds2ldif** utility fails.

All other command-line inputs result in a syntax error message and the correct syntax is displayed. Also, specifying the same option multiple times with different values results in a syntax error.

Examples

1. This example starts the 31-bit version of **ds2ldif** to unload all the entries from the TDBM backend named tdbm1 in the LDAP server configuration file /etc/ldap/ds.conf. The **replicateOperationalAttributes** control is not included in the unloaded entries. The two entries that are in the TDBM backend named tdbm1 are written to data set ADMIN3.LDIF.DATA.

   ```
   ds2ldif31 -j -o "//'ADMIN3.LDIF.DATA'" -n tdbm1 -f /etc/ldap/ds.conf
   ```

   The ADMIN3.LDIF.DATA output data set contains the following:

   ```
   version: 1

   dn: o=tdbm
   objectclass: organization
   objectclass: top
   o: tdbm
   ibm-entryuuid: 3A67E000-2E5C-1876-99D0-402064040959
   aclentry: cn=Anybody:normal:rsc:system:rsc
   aclpropagate: TRUE
   entryowner: CN=ADMIN
   ownerpropagate: TRUE

   dn: cn=entry1,o=tdbm
   objectclass: newPilotPerson
   objectclass: person
   objectclass: top
   cn: entry1
   sn: 1
   uid: entry1
   userpassword:: c2VjcmV0
   ibm-entryuuid: 0C3DE000-F85D-1884-94B3-402084027431
   ```

2. This example starts the 64-bit version of **ds2ldif** to unload all the data at and underneath the o=ldbm subtreeDN entry by performing an **unloadRequest** extended operation over SSL with an LDAP root administrator password of secret. By default, the **replicateOperationalAttributes** control is included in the unloaded entries. The entries are written to file /ldbmdata/ldif.data on the system where the LDAP server is running.

   ```
   ds2ldif64 -o /ldbmdata/ldif.data -s o=ldbm -r -w secret -Z -f /etc/ldap/ds.conf
   ```

   The /ldbmdata/ldif.data output file contains the following unloaded entries:

   ```
   version: 1

   dn: o=ldbm
   control: 1.3.18.0.2.10.19 false:: MIGnMB8KAQAwGgQMY3JlYXRvcnNOYW1lMQoECGNuPWF
    kbWluMDAKAQAwKwQPY3JlYXRlVGltZXN0YW1wMRgEFjIwMDgxMDI0MTY0MDUwLjYzMDg5NFowIAo
    BADAbBA1tb2RpZmllcnNOYW1lMQoECGNuPWFkbWluMDAKAQAwKwQPbW9kaWZ5VGltZXN0YW1wMRg
    EFjIwMDgxMDI0MTY0MDUwLjYzMDg5NFo=
   o: ldbm
   objectclass: organization
   objectclass: top
   ibm-entryuuid: 9A063000-FA92-1901-9FBA-402084027431
   aclentry: cn=Anybody:normal:rsc:system:rsc
   aclpropagate: TRUE
   entryowner: CN=ADMIN
   ownerpropagate: TRUE

   dn: cn=replfilter,o=ldbm
   control: 1.3.18.0.2.10.19 false:: MIGnMB8KAQAwGgQMY3JlYXRvcnNOYW1lMQoECGNuPWF
    kbWluMDAKAQAwKwQPY3JlYXRlVGltZXN0YW1wMRgEFjIwMDkwMTE2MTYxMzI5LjA2NTg4NlowIAo
    BADAbBA1tb2RpZmllcnNOYW1lMQoECGNuPWFkbWluMDAKAQAwKwQPbW9kaWZ5VGltZXN0YW1wMRg
    EFjIwMDkwMTE2MTYxMzI5LjA2NTg4Nlo=
   objectclass: ibm-replicationFilter
   objectclass: top
   ```

```
cn: replfilter
ibm-replicationfilterattr: (objectclass=person):(cn,sn)
ibm-entryuuid: 1014C000-B229-1970-817D-402084027431

dn: ou=rochester,o=ldbm
control: 1.3.18.0.2.10.19 false:: MIGnMB8KAQAwGgQMY3JlYXRvcnNOYW1lMQoECGNuPWF
 kbWluMDAKAQAwKwQPY3JlYXRlVGltZXN0YW1wMRgEFjIwMDkwMTE2MTYwODU5LjM5NjkwOVowIAo
 BADAbBA1tb2RpZmllcnNOYW1lMQoECGNuPWFkbWluMDAKAQAwKwQPbW9kaWZ5VGltZXN0YW1wMRg
 EFjIwMDkwMTE2MTYwODU5LjM5NjkwOVo=
objectclass: organizationalUnit
objectclass: top
ou: rochester
ibm-entryuuid: 60E5C000-B11B-1970-817D-402084027431

dn: cn=entry1,ou=rochester,o=ldbm
control: 1.3.18.0.2.10.19 false:: MIGnMB8KAQAwGgQMY3JlYXRvcnNOYW1lMQoECGNuPWF
 kbWluMDAKAQAwKwQPY3JlYXRlVGltZXN0YW1wMRgEFjIwMDkwMTE2MTYwOTUxLjMxNjM3MVowIAo
 BADAbBA1tb2RpZmllcnNOYW1lMQoECGNuPWFkbWluMDAKAQAwKwQPbW9kaWZ5VGltZXN0YW1wMRg
 EFjIwMDkwMTE2MTYwOTUxLjMxNjM3MVo=
objectclass: newPilotPerson
objectclass: person
objectclass: top
cn: entry1
sn: entry1sn
userpassword:: c2VjcmV0
description: A very nice person
ibm-entryuuid: 4D3BE000-B14F-1970-817D-402084027431
```

3. This example builds on the previous example, however, uses a replication filter entry to filter the unload data. The `cn=replfilter,o=ldbm` replication filter entry from 2 on page 227 has an **ibm-replicationFilterAttr** attribute value that only allows entries that have an **objectclass** value of **person** to be unloaded and only unloads the **cn** and **sn** attributes (if any) from these entries. See "Partial replication" on page 536 for more information about replication filtering.

   ```
   ds2ldif31 -o /ldbmdata/filtered.ldif.data –s o=ldbm -r -w secret –q cn=replfilter,o=ldbm
    -f /etc/ldap/ds.conf
   ```

   The `/ldbmdata/filtered.ldif.data` output file contains the following unloaded entry:

   ```
   version: 1

   dn: cn=entry1,ou=rochester,o=ldbm
   control: 1.3.18.0.2.10.19 false:: MIGnMB8KAQAwGgQMY3JlYXRvcnNOYW1lMQoECGNuPWF
    kbWluMDAKAQAwKwQPY3JlYXRlVGltZXN0YW1wMRgEFjIwMDkwMTE2MTYwOTUxLjMxNjM3MVowIAo
    BADAbBA1tb2RpZmllcnNOYW1lMQoECGNuPWFkbWluMDAKAQAwKwQPbW9kaWZ5VGltZXN0YW1wMRg
    EFjIwMDkwMTE2MTYwOTUxLjMxNjM3MVo=
   objectclass: newPilotPerson
   objectclass: person
   objectclass: top
   cn: entry1
   sn: entry1sn
   ibm-entryuuid: 4D3BE000-B14F-1970-817D-402084027431
   ```

   Note:

   a. The `o=ldbm`, `cn=replfilter,o=ldbm` and `ou=rochester,o=ldbm` entries are not unloaded because they do not have an **objectclass** attribute value of **person**.

   b. The `cn=entry1,ou=rochester,o=ibm` entry is allowed to be unloaded because it has an objectclass attribute value of **person**. However, the **userpassword** and **description** attribute values are filtered out because they are not specified in the replication filter.

c. The **objectclass** and **ibm-entryuuid** attribute values are always unloaded by the **ds2ldif** utility although they are not specified in the replication filter.

Using the -t (tagging) option

When the **-t** option is used on the **ds2ldif** utility, the format of the unloaded **userPassword** or **ibm-slapdAdminPw** attribute depends on how the value is encrypted or hashed.

1. If the value is hashed using crypt, MD5, SHA, SHA224, SHA256, SHA384, or SHA512 one-way hashing algorithms, then the tag is visible and the hashed **userPassword** or **ibm-slapdAdminPw** value is base64 encoded in the unloaded value. The format of the unloaded value is:

   ```
   attrtype: {tag}base64encoded_and_hashedvalue
   ```

   where, *attrtype* is `userpassword` or `ibm-slapdadminpw`. *tag* is `crypt`, `MD5`, `SHA`, `SHA224`, `SHA256`, `SHA384`, or `SHA512`. For example:

   ```
   userpassword: {crypt}0fCik9fUqZnixuKkYQ==
   userpassword: {MD5}34d121/hie8s
   userpassword: {SHA}24309gf[jgt
   userpassword: {SHA224}lcf7ypKsUIOv2mKlZKPQFPw7cskUDjy5nqa/Eg==
   userpassword: {SHA256}K7gNU3sdo+OL0wNhqoVWhr3g6s1xYv72ol/pe/Unols=
   userpassword: {SHA384}WKd1ukESvjAFrkQHznV9iP2nHUBJe7gCbsrFTU4//HIyzo3jq1rLMK4
    5dg/ufFPt
   userpassword: {SHA512}vSsar3708Jvp9Szi2NWZZ02Bqp1qRCFpbcTZPdBhnWgs5WtNZKnvCXd
    hztmeD2cmW192CF5bDufKRpayrW/isg==
   ```

2. If the value is hashed using SSHA (Salted SHA), SSHA224, SSHA256, SSHA384, or SSHA512 hashing algorithms, then the tag is visible and the hashed **userPassword** or **ibm-slapdAdminPw** value and salt values are base64 encoded in the unloaded value. The format of the unloaded value is:

   ```
   attrtype: {tag}base64encoded_hashed_and_salt_values
   ```

   where, *attrtype* is `userpassword` or `ibm-slapdadminpw`. *tag* is `SSHA`, `SSHA224`, `SSHA256`, `SSHA384`, or `SHA512`. For example:

   ```
   userpassword: {SSHA}yEmjV/P1OsnkDbFMpXARCpUzA0evrN4xquMjGW5bMKlhaAkb5Zt6VQ==
   userpassword: {SSHA224}sEHZjTzABWiPMDSFmnuYDUUMzGF1C+XOcryro0otC3X3smmwNlAbs+
    222PJRElE7GJUZ4adtbpo=
   userpassword: {SSHA256} qFzEm0vg2BtJL0cK6baEv6VrRJ4MI+wqQtvoknWjEA5lAL3ePW2u0l
    Ur6q+Ye/UYJG+eOyaAuHEehFN3OkGJwA==
   userpassword: {SSHA384}mbP0pQkuXYlDswEDq6JYWp2Y95jgysAX0wohTmbKP74tQvnkRl9G5e
    u46qth1jOKfvm7HItIuCzcdSRMTe80vynEsv+l0eSfge6Ou3yrXs0cNeN/yw5yMp+FUX0HIg4f
   userpassword: {SSHA512}rR/ls84oX0qz/GuxGsdtKaRwhdBXDVEP3Uj/WIRB+KB7zON8DX48gA
    L1k1QCRnrLv0jvyBEB45Dmyj71Awt3M2T5PeagtoTIxbDs1XgVH7zDqAHosWJEI0ZnOviQFP3Cx6
    lR3OMOtd5XEAJKC3RBTnhYkOXmdqqwe6KkorUdaMQ=
   ```

3. If the value is encrypted using a two-way encryption algorithm (DES or AES) or is not encrypted, then the unencrypted value is base64 encoded in the unloaded value and there is no tag. The format of the unloaded value is:

   ```
   attrtype:: base64encoded_and_unencryptedvalue
   ```

   where, *attrtype* is `userpassword` or `ibm-slapdadminpw`. For example:

   ```
   userpassword:: kfa6903axs
   ```

   This is also the format used when unloading **secretKey, replicaCredentials, ibm-replicaKeyPwd**, and **ibm-slapdMasterPw** attribute values because these values can only be encrypted using two-way encryption algorithms.

Note:

1. The LDAP server loads and uses tagged **userPassword** or **ibm-slapdAdmimPw** values that are hashed in crypt, SHA, SSHA (Salted SHA), or MD5 and were unloaded using the **ds2ldif** utility with the **-t** option. Also, these tagged values

might be acceptable for other LDAP providers to load into their directory. If it is not directly loadable, this format is easily modifiable for loading by another provider into its LDAP directory.

2. The values returned by the **crypt** algorithm are portable only to other X/Open-conformant systems when the **pwCryptCompat** configuration option is set to **off**. When the **pwCryptCompat** configuration option is set to **off**, the **crypt()** algorithm uses ASCII, which is a subset of UTF-8, when generating the hashed **userPassword** or **ibm-slapdAdminPw** attribute values. Therefore, you might want to set the **pwCryptCompat** configuration option to **off** when it is necessary to share **userPassword** or **ibm-slapdAdminPw** attribute values hashed in **crypt()** between the z/OS LDAP server and other ASCII-based LDAP servers. For more information about the **pwCryptCompat** configuration option, see page "pwCryptCompat {on | off}" on page 118.

When the **-t** option is not used on the **ds2ldif** utility, the format of the unloaded password attributes depends upon how the **userPassword** or **ibm-slapdAdminPw** value is encrypted or hashed.

1. If the value is hashed using crypt, SHA, or MD5 one-way hashing algorithms, then the tag and the hashed **userPassword** or **ibm-slapdAdminPw** values are base64 encoded in the unloaded value. The format of the unloaded value is:

   ```
   attrtype:: base64encodedValue
   ```

   where, *attrtype* is `userpassword` or `ibm-slapdadminpw`. *base64encodedValue* is a base64 encoding of *{tag}* `hashedvalue`.

   where, *tag* is `crypt`, `MD5`, or `SHA`. For example:

   ```
   userpassword:: e2NyeXB0fdHwopPX1KmZ4sbipGE=
   ```

2. If the value is hashed using SHA224, SHA256, SHA384, or SHA512 one-way hashing algorithms, then the tag is visible and the hashed **userPassword** or **ibm-slapdAdminPw** value is base64 encoded in the unloaded value. The format of the unloaded value is:

   ```
   attrtype: {tag}base64encoded_and_hashedvalue
   ```

   where, *attrtype* is `userpassword` or `ibm-slapdadminpw`. *tag* is `SHA224`, `SSHA256`, `SSHA384`, or `SHA512`. For example:

   ```
   userpassword: {SSHA224}sEHZjTzABWiPMDSFmnuYDUUMzGF1C+XOcryro0otC3X3smmwNlAbs+
    222PJRElE7GJUZ4adtbpo=
   userpassword: {SSHA256}qFzEm0vg2BtJL0cK6baEv6VrRJ4MI+wqQtvoknWjEA5lAL3ePW2u0l
    Ur6q+Ye/UYJG+eOyaAuHEehFN3OkGJwA==
   userpassword: {SSHA384}mbP0pQkuXYlDswEDq6JYWp2Y95jgysAX0wohTmbKP74tQvnkRl9G5e
    u46qth1jOKfvm7HItIuCzcdSRMTe80vynEsv+l0eSfge6Ou3yrXs0cNeN/yw5yMp+FUX0HIg4f
   userpassword: {SSHA512}rR/ls84oX0qz/GuxGsdtKaRwhdBXDVEP3Uj/WIRB+KB7zON8DX48gA
    L1k1QCRnrLv0jvyBEB45Dmyj71Awt3M2T5PeagtoTIxbDs1XgVH7zDqAHosWJEI0ZnOviQFP3Cx6
    lR3OMOtd5XEAJKC3RBTnhYkOXmdqqwe6KkorUdaMQ=
   ```

3. If the value is hashed using SSHA (Salted SHA), SSHA224, SSHA256, SSHA384, or SSHA512 one-way hashing algorithms, then the tag is visible and the hashed **userPassword** or **ibm-slapdAdminPw** value and salt values are base64 encoded in the unloaded value. The format of the unloaded value is:

   ```
   attrtype: {tag}base64encoded_hashed_and_salt_values
   ```

   where, *attrtype* is `userpassword` or `ibm-slapdadminpw`. *tag* is `SSHA`, `SSHA224`, `SSHA256`, `SSHA384`, or `SSHA512`. For example:

   ```
   userpassword: {SSHA}yEmjV/P1OsnkDbFMpXARCpUzA0evrN4xquMjGW5bMKlhaAkb5Zt6VQ==
   userpassword: {SSHA224}sEHZjTzABWiPMDSFmnuYDUUMzGF1C+XOcryro0otC3X3smmwNlAbs+
    222PJRElE7GJUZ4adtbpo=
   userpassword: {SSHA256}qFzEm0vg2BtJL0cK6baEv6VrRJ4MI+wqQtvoknWjEA5lAL3ePW2u0l
    Ur6q+Ye/UYJG+eOyaAuHEehFN3OkGJwA==
   userpassword: {SSHA384}mbP0pQkuXYlDswEDq6JYWp2Y95jgysAX0wohTmbKP74tQvnkRl9G5e
   ```

```
 u46qth1jOKfvm7HItIuCzcdSRMTe80vynEsv+l0eSfge6Ou3yrXs0cNeN/yw5yMp+FUX0HIg4f
userpassword: {SSHA512}rR/ls84oX0qz/GuxGsdtKaRwhdBXDVEP3Uj/WIRB+KB7zON8DX48gA
 L1k1QCRnrLv0jvyBEB45Dmyj71Awt3M2T5PeagtoTIxbDs1XgVH7zDqAHosWJEI0ZnOviQFP3Cx6
 lR3OMOtd5XEAJKC3RBTnhYkOXmdqqwe6KkorUdaMQ=
```

4. If the value is encrypted using a two-way encryption algorithm (DES or AES) or is not encrypted, then the unencrypted value is base64 encoded in the unloaded value and there is no tag present. The format of the unloaded value is:

   ```
   attrtype:: base64encoded_and_unencryptedvalue
   ```

   where, *attrtype* is `userpassword` or `ibm-slapdadminpw`. For example:

   ```
   userpassword:: e01ENXkfa6903axs
   ```

   This is also the format used when unloading **secretKey, replicaCredentials, ibm-replicaKeyPwd**, and **ibm-slapdMasterPw** attribute values, because these values can only be encrypted using two-way encryption algorithms.

Using the unloadRequest extended operation

The **unloadRequest** extended operation is used to remotely unload directory data from a currently running z/OS LDAP server. The **unloadRequest** extended operation is required when attempting to unload data from an LDBM or CDBM backend that is already running with an active z/OS LDAP server. The **unloadRequest** extended operation is also required when using the **-q** *filterDN* option to filter entries as they are being unloaded from an LDBM, TDBM, or CDBM backend. The **-r** option can be used to force the **unloadRequest** extended operation to be sent to the z/OS LDAP server for any unload operation. If running **ds2ldif** from JCL, a DD card is not allowed to be specified on the **-o** option when using the **unloadRequest** extended operation.

The **ds2ldif** utility does the following when an **unloadRequest** extended operation is performed:

1. An LDAP root administrator simple bind is attempted using each secure **listen** option in the LDAP server configuration file until a successful secure connection is established with the LDAP server. The **sslKeyRingFile** option in the LDAP server configuration file indicates the key database file, RACF key ring, or PKCS #11 token that **ds2ldif** uses to communicate securely with the LDAP server. If the **sslKeyRingFile** option is a key database file, the **sslKeyRingFilePW** or **sslKeyRingPWStashFile** options in the LDAP server configuration file are used by **ds2ldif** to gain access to the key database file. The **sslCertificate** option in the LDAP server configuration file is used as the SSL certificate when **ds2ldif** establishes the secure connection with the LDAP server. The **adminDN** that is specified in the LDAP server configuration file is used as the bind DN. If there is an **adminPW** configuration option present, it is used as the password. If there is no **adminPW** configuration option, **ds2ldif** uses the value of the **ds2ldif –w** option or issues a prompt for the password if the **-w** option is not specified.

   Note:

   a. The **ds2ldif** utility sends the **PasswordPolicy** control as a non-critical control when the LDAP root administrator in the configuration file (**adminDN** configuration option) attempts to authenticate to the LDAP server. If the LDAP root administrator's entry is subject to password policy on the targeted LDAP server, the **ds2ldif** utility parses and displays the warning and error messages from the **PasswordPolicy** control response.

b. If you are running **ds2ldif** from TSO or batch, the password prompt does not execute. In these environments, you must either have an **adminPW** configuration option or specify the **-w** option.
c. The **getpass()** routine used to prompt for the password returns at most PASS_MAX number of characters, truncating any additional characters. See the description of **getpass()** in *z/OS XL C/C++ Runtime Library Reference* for more information. If the length of the root administrator (**adminDN**) password is greater than PASS_MAX, you must either have an **adminPW** configuration option or specify the **-w** option.
d. Ensure that the user ID running the **ds2ldif** utility has access to the key database file, RACF key ring, or PKCS #11 token that is specified on the **sslKeyRingFile** option.

2. If a secure connection is not established with the LDAP server or there are no secure **listen** options in the LDAP server configuration file, an LDAP root administrator simple bind is attempted using each nonsecure **listen** option until a successful nonsecure connection is established. If a connection is not established with the LDAP server, **ds2ldif** ends.

   Note:

   a. To perform the **unloadRequest** extended operation with the **ds2ldif** utility, the bound user is the LDAP root administrator (**adminDN**).
   b. If the **-Z** option is specified, the **ds2ldif** utility communicates over a secure port if a connection is established. If a secure connection is not established, the **ds2ldif** utility fails.

3. Entries are unloaded into an output file on the LDAP servers system. The name of the file is specified in the **ds2ldif -o** option.

   Note: If the **ds2ldif** utility unloads entries when using the **unloadRequest** extended operation, the output file is produced in the code page that is specified in the LANG environment variable of the LDAP server. Otherwise, the output file is produced in the code page that is specified in the LANG environment variable of the **ds2ldif** utility.

4. An **unloadResponse** extended operation is sent back to the **ds2ldif** utility indicating how many entries are unloaded and whether the extended operation is successful.

Additional set up when unloading LDBM or CDBM directory data

If the **ds2ldif** utility is being used to unload LDBM or CDBM directory data and an **unloadRequest** extended operation is not being performed (the LDAP server is not running), the user ID that is running the **ds2ldif** utility must have read and execute access to the directory that is specified on the **databaseDirectory** configuration option of the LDBM or CDBM backend that is being unloaded and must also have read access to the LDBM or CDBM database and checkpoint files in that directory to successfully perform an unload. Also, the user ID must be in the owning GID group or be the owning UID for the directory and files in that directory to successfully perform an unload.

You can use the RACF command in the following example to define the user ID that runs the **ds2ldif** utility (substitute the values that you want for DFLTGRP and UID).

```
ADDUSER LDPADMIN DFLTGRP(LDAPGRP) OMVS(UID(2)
PROGRAM('/bin/sh'))
```

You can use the RACF CONNECT command to add an existing user to a group.

```
CONNECT LDAPADMIN GROUP(LDAPGRP)
```

See "Requirements for a user ID that runs the LDAP server" on page 47 for similar commands to create the user ID and group for the LDAP server. Note that the commands also use LDAPGRP.

If the **ds2ldif** utility is being used to unload directory data and an **unloadRequest** extended operation is not being performed (the LDAP server is not running), the user ID that is running the **ds2ldif** utility must have read and execute access to the directory specified by the **schemaPath** configuration option or its default value and it must also have read access to the schema.db file in the directory. Also, the user ID must be in the owning GID group or be the owning UID for the directory and schema.db file in that directory to successfully perform an unload.

Usage

1. The output file name option (**-o**) is a required option for the **ds2ldif** utility. See "Specifying a value for filename" on page 86 for information about specifying the output file name. If the output is to be written to a z/OS UNIX System Services file system, the file name must be fully qualified. If the output is to be written to a sequential or partitioned data set, the data set should be pre-allocated with a record format of variable blocked (VB), a record length of 1028, and a blocksize of 6144. The size of the output LDIF file is generally smaller than the space consumed in the database for both LDBM and TDBM, with the LDIF file often using less than half the space used in the database. However, unusual cases, such as a high amount of binary data or a high number of multi-valued attributes, can inflate the LDIF file size such that it might be slightly larger than the space used in the database. If you are unloading the entire backend, the size of the pertinent data can be estimated from the TDBM database by adding the SPACEF values (in kilobytes) of the DIR_ENTRY, DIR_LONGENTRY, and DIR_LONGATTR tables from a recent DB2 RUNSTATS report. For LDBM backends, the sum of the file sizes of all the **LDBM-*x*.db** files (where *x* is a number such as 1, 2, 3, and so on) provide a database size estimate. However, if the **LDBM.ckpt** file is large, this might include more data that has yet to be merged into the database files. In this case, you can either include the size of the **LDBM.ckpt** file in the sum or perform an LDBM commit to get a better estimate.
2. If the **ds2ldif** utility is invoked with the **-s** or the **-n** option and there is only one TDBM, LDBM, or CDBM backend in the LDAP server configuration file, all the directory entries (other than **cn=localhost** unless the **-l** option is specified) in that particular TDBM, LDBM, or CDBM backend are unloaded.
3. If a TDBM backend contains entries that have a suffix that is no longer configured in the TDBM section of the LDAP server configuration file, **ds2ldif** unloads those entries when it is invoked with the **-n** option. This is not true for an LDBM backend: only the entries that have a currently configured LDBM suffix are unloaded.
4. The **ds2ldif** utility only unloads owner and ACL information for entries that have a specific owner or ACL. Any entry data with an inherited owner or ACL does not have owner or ACL information unloaded.
5. The **ibm-entryuuid** attribute is included in each entry unloaded by the **ds2ldif** utility. The **ldapadd** utility can be used to add an entry containing an **ibm-entryuuid** attribute if the bound user is an LDAP root administrator and the server is running in maintenance mode or the **-k** option (**Server Administration** server control) is specified in the **ldapadd** utility. The **ldif2ds** utility supports loading entries containing the **ibm-entryuuid** attribute into a

TDBM backend. If you do not need the **ibm-entryuuid** attribute in the loaded entry to have the same value as in the entry that was unloaded, then remove the attribute from each entry in the **ds2ldif** output file and a new **ibm-entryuuid** value is assigned when the entry is loaded using **ldapadd** or **ldif2ds**.

6. The **replicateOperationalAttributes** control is added to each unloaded entry when the **-j** option is not specified and the server compatibility level is 5 or greater. The **replicateOperationalAttributes** control value has the **modifyTimestamp, createTimestamp, creatorsName**, and **modifiersName** attribute types and values for the entry, base64 encoded. Both the **ldapadd** command and the **ldif2ds** utility support the specification of the **replicateOperationalAttributes** control in the input LDIF file. See the **replicateOperationalAttributes** control in "replicateOperationalAttributes" on page 688 for more information about the control. See the **serverCompatLevel** configuration option "serverCompatLevel {3 | 4 | 5 | 6 | 7| 8}" on page 125 for more information about the server compatibility level.
7. If there are password policy operational attribute values present in the entries that are being unloaded, they are included in the LDIF file created by the **ds2ldif** utility. The password policy operational attributes are **pwdChangedTime, pwdAccountLockedTime, pwdExpirationWarned, pwdFailureTime, pwdGraceUseTime, pwdHistory, pwdReset, ibm-pwdAccountLocked, ibm-pwdIndividualPolicyDN**, and **ibm-pwdGroupPolicyDN**.
8. If password policy is active in the LDAP server and there are plans to populate a new LDAP server with the same data, you might want the password policies in the **cn=ibmpolicies** suffix in the CDBM backend be unloaded along with the data you want in the LDBM or TDBM backends. These unloaded password policy entries must be added to the new server under the **cn=ibmpolicies** suffix in case there are user or group entries that have **ibm-pwdgrouppolicydn** or **ibm-pwdindividualpolicydn** attribute values. These user or group entries cannot be added to the server with the **ldapadd** or **ldif2ds** utilities until all referenced password policy entries are added to the CDBM backend.
9. The LDAP Version 3 protocol has a related set of internet drafts that document the introduction of a version mechanism for use in creating LDIF files. The **ds2ldif** 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 that are contained in the version one LDIF file are portable characters that are represented in the local code page that is specified in the LANG environment variable of the **ds2ldif** utility. If the **unloadRequest** extended operation is used, the LDIF file is in the local code page that is specified in the LANG environment variable of the LDAP server. If you reload the data in the LDIF file, ensure the utility that loads the data expects the data in the same code page you used to create the LDIF file. Strings containing nonportable characters (for example, textual values containing multi-byte UTF-8 characters) are base64 encoded.
10. To unload the LDAP server schema entry, specify:

    ```
    -s cn=schema
    ```

    The schema entry is unloaded in LDIF modify format. The current LDAP server schema entry is unloaded. No merging of TDBM schema into the LDAP server schema is performed when running **ds2ldif**.

11. If you want to unload the schema entry from a TDBM backend that is migrated from an Integrated Security Services LDAP server, specify the following:

    ```
    -s cn=schema,suffixDN
    ```

    where *suffixDN* is the DN of a suffix in the TDBM backend. The schema entry is unloaded in LDIF modify format. The current TDBM schema entry is unloaded. No merging of the LDAP server schema into the TDBM schema entry is performed when running **ds2ldif**.
12. When you edit the output file that is produced by **ds2ldif**, 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 results in deleting the blanks, therefore, changing the value of the attribute. This is important when the line is continued, because the existing blanks no longer separate the last word in the line from the continuation on the next line. The maximum line length in a **ds2ldif** output file is 77; continued lines are always 77 long when the file is created by **ds2ldif**. The **oedit** editor is an example of an editor that deletes blanks and is not used.
13. The **LDAP_DEBUG** environment variable can also be used to set the debug level for **ds2ldif**. See debugLevel for more information about specifying the debug level.
14. If running **ds2ldif** in 64-bit mode, ensure that the user ID running the **ds2ldif** utility has a sufficient value specified for the MEMLIMIT and ASSIZEMAX values in the OMVS segment so that storage greater than 2 GB can be used. This enables the **ds2ldif** utility to work properly in 64-bit mode.
15. If you are using **ds2ldif** to unload many entries that you plan to later load using **ldif2ds**, specify the **-g** option on both **ds2ldif** and **ldif2ds**. This causes **ds2ldif** to unload the entries using a depth-first traversal of the directory: the root is unloaded, then each subtree directly below the root is traversed. This order results in improved **ldif2ds** load capacity, at the cost of some affect on **ds2ldif** performance. **ldapadd** performance does not benefit from the **-g** option for **ds2ldif**.

Diagnosis

Exit status is 0 if no errors occur. Errors result in a nonzero exit status and a diagnostic message being written to standard error.

ldif2ds utility

Purpose

The **ldif2ds** utility is used to load entries that are specified in LDAP Data Interchange Format (LDIF) into a TDBM directory that is stored in a relational database. **ldif2ds** cannot be used to load entries into a GDBM, LDBM, CDBM, or SDBM directory. The TDBM directory must exist. The **ldif2ds** utility is intended for loading many 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 directory.

The **ldapadd** utility can also be used to add entries to a TDBM directory. See "When to use ldif2ds" on page 241 for more information about when to use **ldif2ds** or **ldapadd**. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information about **ldapadd**.

The **ldif2ds** utility might be used to add entries to an empty directory or to a directory that already contains entries. The utility cannot be used to modify the LDAP server schema. The schema must already be appropriate for the entries being added.

There are two versions of the **ldif2ds** utility:
- **ldif2ds31** (runs in a 31-bit environment)
- **ldif2ds64** (runs in a 64-bit environment)

See "Preparing to run ldif2ds" on page 238 before using **ldif2ds**.

Format

```
ldif2ds | ldif2ds31 | ldif2ds64 { [ { [-c] [-g]
             [-p [-o outHlq] [-j] [-b creator] [-k keyFile]] }
           { [-i ldifFile[,ldifFile]...] [-e ldifListFile] }
           [-f confFile] [-q summaryFrequency] ]
         [ -l [-o outHlq] ] }
       [-d debugLevel] [-?]
```

Note: **ldif2ds** can be used in place of **ldif2ds31**

Parameters

The following table shows the options that you can use for the **ldif2ds** utility:

Option	Description
-?	Display the usage.
-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, **ldif2ds** fails. The same processing is also applied for the modifier for each loaded entry that does not already include a **modifiersname** attribute. This option is ignored if **ldif2ds** is not invoked for the prepare step.
-c	Check that the entries in the LDIF input file or files are complete and acceptable according to the LDAP server schema. The DB2 database is not modified during the check phase.
-d *debugLevel*	Specify the level of the debug messages to be created. The debugLevel is specified in the same fashion as the debug level for the LDAP server. LDAP debug levels lists the specific debug levels. The default is no debug messages.
-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 **ldifFile option** for more information about using a list of LDIF files. This option is ignored if **ldif2ds** is not invoked for the check or prepare step.

Option	Description
-f *confFile*	Specify the name of the LDAP server 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 configuration data set specified by the CONFIG DD statement is used if the **-f** parameter is not specified. The `/etc/ldap/ds.conf` configuration file is used if the **-f** parameter is not specified and the CONFIG DD statement is not defined. This option is ignored if **ldif2ds** is not invoked for the check or prepare step. See "Specifying a value for filename" on page 86 for information about how to specify the file name.
-g	Specify that the entries to be loaded are in genealogical order in the input LDIF files. This is the order that results from doing a depth-first traversal of a directory. This order greatly reduces the storage usage of **ldif2ds** during the check and prepare steps, enabling more entries to be loaded at one time. Do not specify this option if the entries to load are not in genealogical order. This order is produced when entries are unloaded using **ds2ldif** with the **-g** option.
-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 **ldif2ds** is invoked separately to run the check and prepare steps, make sure to specify the same set of LDIF files each time you run **ldif2ds**. See "Specifying a value for filename" on page 86 for information about how to specify the file name. See the description of the **ldapmodify** utility in the LDAP client utilities chapter in *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information about the general format of LDIF entries. This option is ignored if **ldif2ds** is not invoked for the check or prepare step.
-j	Specify that DB2 logging is used when loading the directory entries. The load operation is faster if DB2 logging is not used but the DB2 database is placed into copy-pending state after the data has been loaded. You then must run either the DB2 **COPY** or **REORG** utility to reset the copy-pending state. **Note:** This option can only be specified during the prepare step, when the JCL for the load jobs is created. It is ignored if **ldif2ds** is not invoked for the prepare step.
-k *keyFile*	Specify the name of the file containing the LDAP server encryption keys. If running **ldif2ds** in batch, the data set specified by the **LDAPKEYS DD** statement is used if the **–k** option is not specified. The key file must be specified if any entries loaded have **userPassword, secretKey, ibm-replicaKeyPwd, ibm-slapdMasterPw**, or **replicaCredentials** attribute values that are encrypted using the DES or AES algorithm and not using ICSF. These attribute values are encrypted as the entry is prepared for loading into the directory.
-l	Start the DB2 Load Utility to load the TDBM database.
-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 data sets, whose high-level qualifier is specified with the **-o** option. The prepare step deletes the existing contents of the data sets before writing new contents to the data sets. The DB2 database is modified during the prepare phase as entry identifiers and attribute identifiers are assigned for the new directory entries. The check phase is always performed when the **-p** option is specified, even if the **-c** option is not specified.

Option	Description
-o *outHlq*	Specify the high-level qualifier of the data sets that contains the DB2 load and JCL output, the status information, and the system information. These data sets must be allocated before invoking **ldif2ds**. The value cannot be more than 22 characters long. See "Preparing to run ldif2ds" for more information. This option is ignored if **ldif2ds** is not invoked for the prepare or load step.
-q *summaryFrequency*	Specify the number of LDIF entries the check or prepare step processes between issuing summary messages. The default value is 1000, resulting in issuing a summary message after every 1000 entries are checked or 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. This option is ignored if **ldif2ds** is not invoked for the check or prepare step.

All other command-line inputs result in a syntax error message and the correct syntax is displayed. Also, specifying the same option multiple times with different values results in a syntax error message. File names are case-sensitive unless the file name refers to a data set or DD statement. A data set is indicated by //'*dsname*' while a DD statement is indicated by //DD:*ddname*.

Examples

Following is an example using the **ldif2ds** utility in 31-bit mode:

```
ldif2ds31 -cpl -i /data2/ldif.data -o admin3.prv -d ERROR
```

This **ldif2ds** utility invocation checks, prepares, and loads the LDIF data from **/data2/ldif.data**, uses the output data sets **ADMIN3.PRV.BULKLOAD.INPUT.xxx** and **ADMIN3.PRV.BULKLOAD.JCL**, and specifies a debug level of ERROR. By default, the server configuration file that is used is `/etc/ldap/ds.conf`. Also, the value of the **adminDN** option in the LDAP server configuration file is used as the default creator of each loaded entry, no DB2 logging is done when loading the entries, and a progress message is issued after every 1000 entries processed during the check and prepare steps.

Following is an example using the **ldif2ds** utility in 64-bit mode:

```
ldif2ds64 -cpl -i "//'ADMIN3.LDIF.DATA'" -o admin3.prv -d ERROR
```

Preparing to run ldif2ds

Before invoking **ldif2ds**, you must:

- Allocate the output data sets used by **ldif2ds**.
- Create the SYSTEM file used by the prepare step of **ldif2ds**.
- Set up the user ID to run **ldif2ds** to ensure sufficient access to directories and files.

Setting up a user ID to run ldif2ds

The **ldif2ds** utility accesses the schema directory and file, and the CDBM backend directory and files. The user ID that runs **ldif2ds** must have the following attributes:

- The user ID must have read and execute access to the directory specified by the schemaPath Directory server configuration option or its default value and read access to the schema.db file in that directory.

- The user ID must be in the owning GID group or be the owning UID for the directory and schema.db file in that directory.
- The user ID must have read access to the CDBM backend directory as specified by the databaseDirectory server configuration option or its default value and the files within that directory.
- The user ID must be in the owning GID group or be the owning UID for the CDBM backend directory and files within that directory.

You can use the RACF commands in the following example to define the user ID that runs the **ldif2ds** utility (substitute appropriate values for DFLTGRP and UID).

```
ADDUSER LDPADMIN DFLTGRP(LDAPGRP) OMVS(UID(2)
PROGRAM('/bin/sh'))
```

You can use the RACF CONNECT command to add an existing user to a group.

```
CONNECT LDAPADMIN GROUP(LDAPGRP)
```

See "Requirements for a user ID that runs the LDAP server" on page 47 for similar commands to create the user ID and group for the LDAP server. Note that the commands also use LDAPGRP.

Allocating data sets required by ldif2ds

The various **ldif2ds** processing steps use six load data sets and a JCL data set. These data sets must be allocated before invoking **ldif2ds** and the high-level qualifiers in the data set names must be specified as the value of the **-o** *outHlq* option on the command. The **ldif2ds** utility writes the contents of these data sets, except for the SYSTEM member of the JCL data set which must be created before **ldif2ds** is run.

- Load record data sets

  The prepare step of **ldif2ds** writes the load data created from each LDIF entry into the load record data sets. Each data set contains the records for one database table and is used as input to the DB2 Load Utility when loading that table.

  – Data set names:
    outHlq.BULKLOAD.INPUT.DESC - DIR_DESC load data set
    outHlq.BULKLOAD.INPUT.ENTRY - DIR_ENTRY load data set
    outHlq.BULKLOAD.INPUT.LATTR - DIR_LONGATTR load data set
    outHlq.BULKLOAD.INPUT.LENTRY - DIR_LONGENTRY load data set
    outHlq.BULKLOAD.INPUT.REPLICA - DIR_REPLICA load data set
    outHlq.BULKLOAD.INPUT.SEARCH - DIR_SEARCH load data set

  – Data set format:
    Sequential (non-PDS)
    Record format = VB

  – Data set record length and block size:

    The record length depends on the page size of the corresponding table space. The actual record length might be reduced slightly by **ldif2ds** at run time.

    For a 32 K page size in DB2, use LRECL=32756, BLKSIZE=32760.

    This record length and block size can also be used 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 enables two blocks for each track and, in general, more bytes for each track to be written. For the smaller page sizes, the LRECL and BLKSIZE must be at least pagesize + 6 and pagesize + 10, in that order.

– Data set size:

A rough estimate for the size (in bytes) of each data set is as follows:

- *outHLQ*.BULKLOAD.INPUT.DESC:

  20 * (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 DNs 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 is written in the *outHLQ*.BULKLOAD.INPUT.ENTRY data set. Otherwise, you might want to allocate each of these three data sets as this maximum size to ensure that you do not run out of space.

- *outHlq*.BULKLOAD.INPUT.REPLICA:

  (number of replicas defined in the LDIF input files) * 24
  If there are no replica entries defined in the LDIF input files, use a minimum size to allocate the data set, such as one block.

- *outHLQ*.BULKLOAD.INPUT.SEARCH:

  (number of bytes in the LDIF input files) * 2.5

• JCL data set

The JCL data set is a PDS whose members contain system information, status information, and the JCL to run DB2 Load Utility for each database table that must be loaded. The prepare and load steps of **ldif2ds** use the status information. The prepare step writes the contents of the JCL members of the JCL data set, using the information in the system member. The system member of the JCL data set must be created by the user before **ldif2ds** is invoked to run the prepare step.

– Data set 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(JREPL) - DIR_REPLICA load JCL
  outHlq.BULKLOAD.JCL(JSRCH) - DIR_SEARCH load JCL
  outHlq.BULKLOAD.JCL(STATUS) - Status information
  outHlq.BULKLOAD.JCL(SYSTEM) - System information - see below

– Data set format:
  Partitioned (PDS)
  Record format = FB
  Record length = 80

– Data set size:
  200 K bytes

Creating the SYSTEM member

The contents of the SYSTEM member, *outHlq*.BULKLOAD.JCL(SYSTEM), must be created before invoking **ldif2ds** to run the prepare step, which uses the information in the SYSTEM member to create the JCL to start the DB2 Load Utility to load each of the database tables.

- Format of SYSTEM records

```
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 or more JOBCARD records. The first JOBCARD record must contain the job name. The JOB statement might span multiple JOBCARD records and must follow the JCL continuation rules. The maximum length of each JCL record is 72 characters. JES control statements might also be included in the JOBCARD specification. For example:

```
JOBCARD //DB2TDBMn JOB (ACCOUNT),PGMR,CLASS=A,MSGCLASS=A,MSGLEVEL=1,
JOBCARD // NOTIFY= &SYSUID
JOBCARD /*JOBPARM SYSAFF=SYS1
```

  The HLQ record is optional. The DB2 high-level qualifier defaults to DSN if the HLQ record is not specified. If specified, the maximum length of the HLQ value is 35.

  Make sure that there are no sequence numbers at the end of each line of the SYSTEM member.

- To differentiate the load jobs in the 6 JCL members of the JCL PDS, **ldif2ds** replaces the last character of the job name with the digits 1 through 6, 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 6 - JCL for loading DIR_REPLICA table, in *outHlq*.BULKLOAD.JCL(JREPL)
  Jobname ends in 5 - JCL for loading DIR_SEARCH table, in *outHlq*.BULKLOAD.JCL(JSRCH)

When to use ldif2ds

Both the **ldif2ds** and **ldapadd** utilities 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 help determine which method to use when loading entries.

Table 25. Considerations for using ldif2ds or ldapadd

Considerations	ldif2ds	ldapadd
Speed of load	Faster	Slower, especially if the database is already large.

Table 25. Considerations for using ldif2ds or ldapadd (continued)

Considerations	ldif2ds	ldapadd
Operational attributes	Accepts input containing operational attributes such as **creatorsname, modifiersname, createtimestamp, modifytimestamp**, and **ibm-entryuuid**.	Input can contain these operational attributes however the **-k** option (**Server Administration** server control) must be specified to allow operational attributes to be added.
Complexity	High, takes time to learn. Typical usage requires multiple invocations, with review of JCL and preparation for recovery before load.	Low.
Set up	User must allocate data sets and create the SYSTEM information file before running **ldif2ds** for the first time.	No setup.
LDAP server downtime	LDAP server must be down during load step. Server can be up during check and prepare 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 performed.	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 that they can be re-used to reload this system.	No saved output.
Replication	New entries are not replicated.	New entries can be replicated.

Summary: The **ldif2ds** utility usage is complex, but it is fast. The **ldapadd** utility usage is simpler, but it is slower.

Suggested: For one-time additions of 100K or more entries, or for frequent additions of 10K or more entries, use **ldif2ds**. For infrequent additions of less than 10K entries, use **ldapadd**. For additions of between 10K and 100K entries, use either **ldif2ds** or **ldapadd**.

ldif2ds performance considerations

The **ldif2ds** utility performance considerations are:

1. **ldif2ds** uses much storage when adding many entries. Make sure that you have sufficient memory available.
2. If the entries to load are in genealogical order, **ldif2ds** can take certain shortcuts to greatly reduce the storage used during the check and prepare steps. This allows **ldif2ds** to process more entries at one time. Genealogical order is the order that results from doing a depth-first traversal of a directory: the root is

visited, then each subtree directly below the root is traversed. Specify the **-g** option to indicate that the LDIF input is in this order. The **-g** option of the **ds2ldif** utility can be used to unload entries in genealogical order. Do not specify the **-g** option on **ldif2ds** if the input is not in genealogical order.

3. If the parent of an entry being added is in the TDBM database, then **ldif2ds** must also check the database to ensure that the entry itself does not exist. Therefore, to minimize the database checks, include the parents of the entries being added in the **ldif2ds** input whenever possible. For example, do not use **ldapadd** to add a suffix and then use **ldif2ds** to add all the entries under the suffix. Instead, include the suffix in the **ldif2ds** input.
4. Do not specify the debug command-line option, **-d**. Usage of debug might affect performance even when there is little debug output. Only specify **-d** when an error occurs that you cannot fix.
5. By default, for better load performance **ldif2ds** 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 sets 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 many entries), change **LOG NO** to **LOG YES** in the JCL. This is done by **ldif2ds** during the prepare step if the **-j** option is specified along with the **-p** option. It can also be performed manually in each of the *outHLQ*.BULKLOAD.JCL members after the prepare step if the load step is run separately from the prepare step.

ldif2ds normal usage

The typical usage of **ldif2ds** is:

1. Perform the necessary setup for running **ldif2ds**, as described in "Preparing to run ldif2ds" on page 238. This consists of:
   - allocating data sets required by **ldif2ds**
   - creating the SYSTEM member in the *outHLQ*.BULKLOAD.JCL data set

   Note the same data sets can be used for loading different entries, but the contents of the data sets are overwritten each time.
2. Repeatedly invoke **ldif2ds** with only the prepare (**-p**) option until all problems in the LDIF input files have been resolved. The check step and the prepare step are performed. You can use the check option (**-c**) instead if you only want to check the entries at this time, then later invoke **ldif2ds** again with the **-p** option.
3. Run **ldif2ds** 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_ENTRY, DIR_LONGATTR, DIR_LONGENTRY, DIR_REPLICA, and the DIR_SEARCH tables. A full image copy is required to help recover from any potential DB2 Load Utility failures. It is performed after the **ldif2ds** invocation with the prepare (**-p**) option specified because the prepare (**-p**) option might update some database tables. Therefore, 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 is performed before the **ldif2ds** invocation.
6. Review the JCL created by the prepare step in the JCL data set, *outHLQ*.BULKLOAD.JCL. Ensure that the DB2 Load Utility data sets are large enough and that the DB2 Load Utility options, especially the **LOG** value, are

acceptable. See note 5 on page 243 in the "ldif2ds performance considerations" on page 242 section for more information about the **LOG** value.

7. Run **ldif2ds** once more, with the load (**-l**) option to load the data into the database, and six batch jobs are submitted to load the database.
8. Review the output from the load jobs when they end. If the loaded table spaces are in the copy pending state, see the *DB2 Utility Guide and Reference* book in IBM Information Management Software for z/OS Solutions Information Center for instructions on removing that restriction on table spaces. This typically 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. See Chapter 31, "Performance tuning," on page 599 for detailed information about running the DB2 **RUNSTATS** utility.
10. Run the DB2 **COPY** utility to make a backup image copy of the database.
11. If running the **ldif2ds** utility in 64-bit mode, ensure that the user ID running the **ldif2ds** utility has a sufficient value specified for the MEMLIMIT and ASSIZEMAX values in the OMVS segment so that storage greater than 2 gigabytes can be used. This enables the **ldif2ds** utility to work properly in 64-bit mode.

ldif2ds recovery

The **ldif2ds** utility can determine whether the submission of the DB2 Load Utility jobs is successful, but it cannot determine if the DB2 Load Utility jobs themselves succeed. In fact, **ldif2ds** typically terminates before the jobs are finished. Therefore, the final **ldif2ds** success message displays even if one or more of the jobs eventually fails.

If a DB2 Load Utility job fails, stop all 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 might result in unnecessary reloading of data. This alternative must be used for recovery when:

- The **ldif2ds** utility is run with the load (**-l**) option immediately after creating the TDBM database (that is, without first starting the server or running **ldif2ds** with the check (**-c**) or prepare (**-p**) options).
- Loading the DIR_ENTRY, DIR_LONGENTRY, or DIR_LONGATTR 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 six table spaces, use the DB2 Recover Utility to recover all six table spaces from those full image copies. Otherwise, drop and re-create the DIR_DESC, DIR_ENTRY, DIR_LONGATTR, DIR_LONGENTRY, DIR_REPLICA, and DIR_SEARCH tables.
2. See the *DB2 Utility Guide and Reference* in IBM Information Management Software for z/OS Solutions Information Center for information about correcting the problems logged in the failing jobs.

3. Run **ldif2ds** 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 **ldif2ds** utility is invoked with the load (**-l**) option immediately after creating the TDBM database.
- Loading the DIR_ENTRY, DIR_LONGATTR, or DIR_LONGENTRY 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_DESC or DIR_SEARCH 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 re-create the failing tables.
  2. See IBM Information Management Software for z/OS Solutions Information Center for information about correcting problems logged in the failing jobs.
  3. Manually resubmit the DB2 Load Utility jobs that failed.
- If the load jobs that failed involve the DIR_ENTRY, DIR_LONGATTR, or DIR_LONGENTRY 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 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.
  2. See IBM Information Management Software for z/OS Solutions Information Center for information about correcting problems logged in the failing jobs.
  3. Manually resubmit the DB2 Load Utility jobs for all the affected tables.

Usage

1. All input files specified with the **-i** and **-e** options can be z/OS UNIX file system files or data sets. To use data sets with **ldif2ds**, you must use a VB record format. Further, separator lines between directory entries in the LDIF file must contain no characters.
2. The **ldif2ds** 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 **ldif2ds**. See debugLevel for more information about specifying the debug level.
4. Run the DB2 **RUNSTATS** utility when the data is loaded so that DB2 queries are optimized. See Chapter 31, “Performance tuning,” on page 599 for detailed information about running the DB2 **RUNSTATS** utility.
5. No replication is performed for entries added by **ldif2ds**. Advanced and basic replication entries can be added using **ldif2ds**, but replication does not begin until the LDAP server is started.
6. Referral entries can be added by **ldif2ds**.
7. Only one TDBM database is loaded in an invocation of **ldif2ds**. 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. **ldif2ds** cannot be used to modify the LDAP server schema. The schema must already contain all the attributes and object classes used by the entries being added.
9. Do not specify the **aclSource** and **ownerSource** attributes in an LDIF entry because they are ignored. These attributes are set only by the system.
10. The **ldif2ds** utility check processing is terminated after 100 syntax errors are detected. The **ldif2ds** prepare and load processing are terminated after the first error. These values cannot be modified.
11. Because typical **ldif2ds** usage can involve multiple invocations to check, prepare, and load the entries, **ldif2ds** 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 **ldif2ds** with the same value for **-o** is considered a continuation of processing of an earlier invocation.

    The status file contains a `STATUS` record and a `TIME` record. When the prepare (**-p**) step is successful, the status is changed to `STATUS P` and the `TIME` record is set to the current time. The load step (**-l**) checks the `STATUS` record and exits if the status is not `P`. When the load step is successful, the status is changed to `STATUS L` and the `TIME` record is reset to the current time.
12. The prepare step uses the current LDAP server schema to create the load records for the entries to be added. Do not change the schema between the prepare step and the load step. This can result in loading entries that might not be valid for the new schema.
13. During the prepare (**-p**) step, 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 **ldif2ds** utility encrypts or hashes clear text **userPassword, secretKey, ibm-replicaKeyPwd, ibm-slapdAdminPw, ibm-slapdMasterPw** and **replicaCredentials** attribute values for new entries loaded into the TDBM backend with the **pwEncryption** and **secretEncryption** methods specified in the LDAP server configuration file. The **ldif2ds** utility loads the LDIF format of an encrypted or hashed value unloaded by the **ds2ldif** utility with or without the **-t** option. When using the **ldif2ds** utility to load entries that have **userPassword** or **ibm-slapdAdminPw** attribute values hashed with crypt, the **pwCryptCompat** configuration option must be set in the same manner as when the values were originally hashed. The **ldif2ds** utility expects that textual data contained within the LDIF file is portable and of UTF-8 origin.
15. If you are maintaining multiple identical TDBM databases, you might want to run the prepare (**-p**) and load (**-l**) steps of **ldif2ds** against each TDBM database. Do not try to use the load data created by running the **ldif2ds** utility prepare step against one database to load entries on a different database. This can result in an unusable TDBM database.
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.

17. When LDIF input contains a **replicateOperationalAttributes** control and the control has a **creatorsname, modifiersname, createtimestamp**, or **modifytimestamp** attribute value, that attribute value is assigned for the attribute in the entry being loaded. An attribute value assigned from the **replicateOperationalAttributes** control replaces any other assigned value for that attribute described in the previous usage note.
18. If there are user or group entries that have **ibm-pwdgrouppolicydn** or **ibm-pwdindividualpolicydn** attribute values specified, the password policy entries referenced in these values must exist in the CDBM backend under the **cn=ibmpolicies** suffix before using the check (**-c**) step of the **ldif2ds** utility. Before running **ldif2ds**, the server must be started with the CDBM backend configured and the server compatibility set to 6 and the necessary password policy entries must be added. When the password policy entries are added to the CDBM backend, shut down the server and then the **ldif2ds** utility can be run.

Diagnosis

Exit status is 0 if no errors occur. Errors result in a nonzero exit status and a diagnostic message being written to standard error.

ldapdiff utility

Purpose

The **ldapdiff** utility is provided to compare two directory subtrees or two schemas on two different directory servers to determine if their contents match. The utility runs as a client operation when both servers are active and determines the differences between the servers. The utility can optionally attempt to synchronize one directory subtree to match the contents of the other.

Format

```
ldapdiff { [-S] [-b baseDn] } [-a] [-C countNumber] [-d debugLevel] [-F] [-j] [-L fileName]
          [-n] [-O] [-v] [-x]
          -sh host [-sp port] [-sD bindDn -sw bindPwd]
                   [-sT truststore -st truststoreType [-sY truststorePwd]
                   [-sZ] [-sK keystore -sN keystoreType [-sP keystorePwd]]]
          -ch host [-cp port] [-cD bindDn -cw bindPwd]
                   [-cT truststore -ct truststoreType [-cY truststorePwd]
                   [-cZ] [-cK keystore -cN keystoreType [-cP keystorePwd]]]
```

Parameters

The following table shows the general options that you can use for the **ldapdiff** utility.

Table 26. ldapdiff general options

Option	Description
-?	Display the usage.

Table 26. ldapdiff general options (continued)

Option	Description
-a	Specify that the **Server Administration** control is to be sent to the consumer server or written to the output LDIF file. This control enables a server that normally refuses updates, such as a quiesced or replica server, to allow updates. If the **-F** option is specified, the control is sent to the consumer server. If the **-L** option is specified, the control is written to the output LDIF file.
-b *baseDn*	Indicate the starting point for the subtree comparison. The **ldapdiff** utility requires either the **-b** or **-S** option, or both.
-C *countNumber*	Specify the maximum number of non-matching subtree entries that can be found between the two LDAP servers. If the number of non-matching subtree entries exceeds the specified number, the **ldapdiff** utility exits.
-d *debugLevel*	Specify the level of debug messages to be created. The default is no debug messages. Currently, only the **ALL** level is supported by the **ldapdiff** utility. This results in a large amount of debug output being created.
-F	Specify that the content on the consumer replica server is modified to match the content of the supplier server. The **-F** option has no effect when doing schema comparisons as requested by the **-S** option.
-j	Indicate that the four operational attributes **creatorsName, createTimeStamp, modifiersName**, and **modifyTimeStamp** are to be ignored when comparing subtrees.
-L *fileName*	Specify the name of the output LDIF file to contain the differences between the supplier and consumer servers. The output LDIF file can be used to update the consumer server to eliminate the differences between the supplier and consumer. The name cannot be a data set.
-n	Specify that the **Do Not Replicate** control is to be sent to the consumer server or written to the output LDIF file. This control prevents the consumer server from sending replicated entries to the next tier of advanced replication servers. If the **-F** option is specified, the control is sent to the consumer server. If the **-L** option is specified, the control is written to the output LDIF file.
-O	Indicate that directory subtrees are compared using checksum only. This option can be used to quickly detect differences between servers without the need to retrieve all attribute types and values. DNs for non-matching entries between the supplier and consumer servers are displayed, but not the non-matching attributes. Both servers must support entry checksum. Otherwise, the **ldapdiff** utility does not start. The **-O** option overrides the **-F, -j**, and **-L** options.
-S	Specify that the schema is to be compared between the supplier and consumer servers. The **ldapdiff** utility requires either the **-b** or **-S** option, or both.
-v	Use verbose mode, with many diagnostics written to standard output.
-x	Specify that extra entries on the consumer server are to be ignored.

The following options apply to the supplier server and are denoted by an initial 's' in the option name:

Table 27. ldapdiff supplier server options

Option	Description
-sD *bindDn*	Specify the DN to use to bind to the supplier server. The *bindDn* parameter should be a string-represented DN. The default is an empty string.
-sh *host*	Specify the host name or IP address on which the supplier server is running. There is no default host, therefore, this option is required.

-sK *keystore*	Specify the file name of the Java keystore or RACF key ring that contains the client certificate. If the supplier server is configured to perform server authentication only, a client certificate is not required. If the supplier server is configured to perform client and server authentication, a client certificate might be required. The file name cannot be a data set.
	
	If specifying a RACF key ring, use the following format:
	`safkeyring://userid/keyFile`
	
	where, *userid* is the RACF user ID that owns the RACF key ring and *keyFile* is the name of the RACF key ring.
	
	If the user ID running **ldapdiff** is the owner of the RACF key ring, then a simplified format might be used:
	`safkeyring:///keyFile`
	
	where, *keyFile* is the name of the RACF key ring.

-sN *keystoreType*	Specify the type of the keystore being used by the **-sK** option. The supported keystore types are: **JCERACFKS** (RACF key ring) **JCEKS** (Java keystore) This is a required option if the **-sK** option is specified. This option cannot be specified by itself.
-sp *port*	Specify the TCP port where the supplier server is listening. The default LDAP non-secure port is 389 and the default LDAP secure port is 636.
-sP *keystorePwd*	Specify the keystore password. This password is required to access the encrypted information in the keystore file, which might include one or more private keys. If the keystore specified by **-sK** is a RACF key ring, then this option should not be specified. This option is not allowed if the **-sK** option is not specified.
-st *truststoreType*	Specify the type of the truststore used by the **-sT** option. The supported truststore types are: **JCERACFKS** (RACF key ring) **JCEKS** (Java keystore) This is a required option if the **-sT** option is specified. This option cannot be specified by itself.

Table 27. ldapdiff supplier server options (continued)

Option	Description

-sT *truststore*	Specify the file name of the Java keystore or RACF keying that contains the CA (certificate authority) certificate that signed the supplier server's certificate.
	
	The file name cannot be a data set.
	
	If specifying a RACF key ring, use the following name format:
	`safkeyring://userid/keyFile`
	
	where, *userid* is the RACF user ID that owns the RACF key ring and *keyFile* is the name of the RACF key ring.
	
	If you have ownership of the RACF key ring, then a simplified format might be used:
	`safkeyring:///keyFile`
	
	where, *keyFile* is the name of the RACF key ring.
	
	This option is required when using SSL. This option enables a secure connection even if the **-sZ** option is not specified.

-sw *bindPwd*	Specify the bind password for simple authentication. The default is an empty string.
-sY *truststorePwd*	Specify the truststore password. This password is required to access the encrypted information in the truststore file, which might include one or more private keys. If the truststore specified by **-sT** is a RACF key ring, then this option should not be specified. This option is not allowed if the **-sT** option is not specified.
-sZ	Use a secure connection to communicate with the supplier server. Secure connections expect the communication to begin with the SSL/TLS handshake.

The following options apply to the consumer server and are denoted by an initial 'c' in the option name:

Table 28. **ldapdiff** *consumer server options*

Option	Description
-cD *bindDn*	Specify the DN to use to bind to the consumer server. The *bindDn* parameter should be a string-represented DN. The default is an empty string.
-ch *host*	Specify the host on which the consumer server is running. There is no default host, therefore, this option is required.

Table 28. **ldapdiff** *consumer server options (continued)*

Option	Description

-cK *keystore*	Specify the file name of the Java keystore or RACF key ring that contains the client certificate. If the consumer server is configured to perform server authentication only, a client certificate is not required. If the consumer server is configured to perform client and server authentication, a client certificate might be required. The file name cannot be a data set.
	
	If specifying a RACF key ring, use the following name format:
	`safkeyring://userid/keyFile`
	
	where, *userid* is the RACF user ID that owns the RACF key ring and *keyFile* is the name of the RACF key ring.
	
	If you have ownership of the RACF key ring, then a simplified format might be used:
	`safkeyring:///keyFile`
	
	where, *keyFile* is the name of the RACF key ring.

-cN *keystoreType*	Specify the type of the keystore being used by the **-cK** option. The supported keystore types are: **JCERACFKS** (RACF key ring) **JCEKS** (Java keystore) This is a required option if the **-cK** option is specified. This option cannot be specified by itself.
-cp *port*	Specify the TCP port where the consumer server is listening. The default LDAP non-secure port is 389 and the default LDAP secure port is 636.
-cP *keystorePwd*	Specify the keystore password. This password is required to access the encrypted information in the keystore file, which might include one or more private keys. If the keystore specified by **-cK** is a RACF key ring, then this option should not be specified. This option is not allowed if the **-cK** option is not specified.
-ct *truststoreType*	Specify the type of the truststore used by the **-cT** option. The supported truststore types are: **JCERACFKS** (RACF key ring) **JCEKS** (Java keystore) This is a required option if the **-cT** option is specified. This option cannot be specified by itself.

Table 28. **ldapdiff** *consumer server options (continued)*

Option	Description

-cT *truststore*	Specify the file name of the Java keystore or RACF keying that contains the CA (certificate authority) certificate that signed the consumer server's certificate.
	
	The file name cannot be a data set.
	
	If specifying a RACF key ring, use the following name format:
	`safkeyring://userid/keyFile`
	
	where, *userid* is the RACF user ID that owns the RACF key ring and *keyFile* is the name of the RACF key ring.
	
	If you have ownership of the RACF key ring, then a simplified format might be used:
	`safkeyring:///keyFile`
	
	where, *keyFile* is the name of the RACF key ring.
	
	This option is required when using SSL. This option enables the **-cZ** option.

-cw *bindPwd*	Specify the bind password for simple authentication. The default is an empty string.
-cY *truststorePwd*	Specify the truststore password. This password is required to access the encrypted information in the truststore file, which might include one or more private keys. If the truststore specified by **-cT** is a RACF key ring, then this option should not be specified. This option is not allowed if the **-cT** option is not specified.
-cZ	Use a secure connection to communicate with the consumer server. Secure connections expect the communication to begin with the SSL/TLS handshake.

All other command-line inputs result in a syntax error message and the correct syntax is displayed. Also, specifying the same option multiple times results in the last specified option being used.

Examples

Assume that two servers are set up, one as a supplier server on IP address `1.2.3.4` and other as a consumer server on IP address `1.2.3.5`, and that the suffix `o=sample` is present on both servers. Assume the entries in the supplier.ldif file are in the supplier server and the entries in the consumer.ldif file are in the consumer server.

- **Contents of supplier.ldif file:**

```
dn: cn=Entry1,o=sample
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
objectclass: ePerson
sn: entry1
cn: Entry1

dn: cn=Entry2,o=sample
objectclass: inetOrgPerson
objectclass: organizationalPerson
```

```
objectclass: person
objectclass: top
objectclass: ePerson
sn: entry2
cn: Entry2
```

- **Contents of consumer.ldif file:**

```
dn: cn=Entry2,o=sample
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
objectclass: ePerson
sn: abcd
cn: Entry2

dn: cn=Entry3,o=sample
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
objectclass: ePerson
sn: entry3
cn: Entry3
```

The following command runs the **ldapdiff** utility to modify the contents of the `o=sample` subtree on the consumer server (even if it is quiesced) to match the supplier server, binding to the supplier server on IP address `1.2.3.4` with a bind DN of `cn=admin` and a password of `supplier` and binding to the consumer server on IP address `1.2.3.5` with a bind DN of `cn=admin` and a password of `consumer`:

```
ldapdiff -b o=sample -sh 1.2.3.4 -sD cn=admin -sw supplier -ch 1.2.3.5 -cD cn=admin
 -cw consumer -F -a
```

The resulting actions are:

1. The `cn=Entry1,o=sample` is added to the consumer server. This entry is already present on the supplier server but was not on the consumer server.
2. The `cn=Entry2,o=sample` is modified on the consumer server. The *sn* attribute value on the `cn=Entry2,o=sample` entry on the consumer server gets modified to match the value on the supplier server.
3. The `cn=Entry3,o=sample` is deleted from the consumer server. This entry does not exist on the supplier server, and is deleted from the consumer server so that the same set of entries are present on both the supplier and consumer servers.

Usage

1. The **ldapdiff** utility is a Java-based utility and requires IBM 31-bit SDK for z/OS, Java 2 Technology Edition, V7.1 (5655-W43) or later to be installed on the system.
2. The **ldapdiff** utility is used to bring a supplier and a consumer server in sync before starting advanced replication. The **ldapdiff** utility requires that the base DN, which is being compared, exist on at least one of the servers.
3. The **ldapdiff** utility is a diagnostic and corrective tool. It is not designed to be run as routine maintenance. If there are out of sync replication-related errors observed, an LDAP administrator with the appropriate administrative roles might decide to run the utility to correct the out of sync condition.
4. When the **ldapdiff** utility is run, do not make any updates to either of the targeted LDAP servers. If the **ldapdiff** utility is run while updates are being made, it cannot be guaranteed that all discrepancies are accurately reported or

fixed. An LDAP root, directory data, or replication administrator must manually quiesce or suspend all update activity to the two subtrees being compared on the two servers. The **ldapdiff** utility does not check at initialization time to determine if the servers are quiesced or not. The replication context subtree can be quiesced by using the **Quiesce or unquiesce context** extended operation on the **ldapexop** utility. See "ldapexop utility" on page 255 for information about the **Quiesce or unquiesce context** extended operation.

5. Use the fix option **(-F)** with caution because it automatically synchronizes all the entries from the supplier server to the consumer server starting from the *baseDn*. This synchronization can result in unintentional results on the consumer server such as the deletion of replication topology entries, RACF users, groups, user-group connections, resource profiles, and other entries. Use the **-L** option first to write any differences between the supplier and consumer servers in LDIF file format. An LDAP administrator analyzes the output LDIF file to ensure that the updates are correct. If they are correct, the changes can be manually applied to the consumer server using the **ldapmodify** command or the **ldapdiff** utility can be rerun with the fix option (**-F**) specified.

   If the **-F** option is specified, use the **Server Administration** server control option (**-a**) to enable the **ldapdiff** utility to write to a read-only replica. The **Server Administration** server control also allows **ldapdiff** to modify operational attributes such as **aclPropagate, aclSource, createTimeStamp, creatorsName, entryOwner, ibm-entryuuid, ibm-pwdGroupPolicyDN, modifiersName, modifyTimeStamp, ownerPropagate, ownerSource, pwdAccountLockedTime, pwdChangedTime, pwdExpirationWarned, pwdFailureTime, pwdGraceUseTime, pwdHistory, pwdReset, ibm-pwdAccountLocked**, and **ibm-pwdIndividualPolicyDN**.

6. The **ldapdiff** utility performs two passes to get the supplier and consumer servers back in sync.

   In the first pass, the **ldapdiff** utility traverses the supplier server and does the following:

   - Checks for any extra entries on the supplier server that are not on the consumer server. The extra entries are added to the consumer server if the **-F** option is specified.
   - Compares the entries that exist on both the supplier and consumer servers. If there are mismatches, the consumer server entries are modified to match the supplier server entries if the **-F** option is specified.

   In the second pass, the **ldapdiff** utility traverses the consumer server to check for any extra entries on the consumer server. The extra entries are deleted from the consumer server if the **-F** option is specified. The second pass can be skipped by specifying the **-x** option.

7. The supplier and consumer bind DNs (**-sD** and **-cD** options) must be able to read the attributes in the entries being compared. In addition, if the fix option (**-F**) is specified, the consumer bind DN must also be able to update attributes and add and delete entries.

8. The **ldapdiff** utility traverses each entry in the directory subtree on the supplier server and compares its contents with the corresponding entry on the consumer server. Because each entry must be retrieved, running the **ldapdiff** utility can take a long time and can generate lots of read requests to the supplier and consumer servers. Depending on how many differences are found and whether the fix option (**-F**) is specified, the **ldapdiff** utility can also generate an equal amount of update (for example add, delete, modify) requests to the consumer server. Use the **ldapdiff** utility once between servers, when replication is initially setup. For example, if your replication topology

has two peer masters and two replica servers, you might want to run the **ldapdiff** utility between peer 1 and peer 2. Then, if replication is suspended, run **ldapdiff** concurrently between peer 1 and replica 1 and between peer 2 and replica 2. If replica servers are out of sync with their master servers, the **ldapdiff** utility can be run to identify and correct out of sync conditions.

9. The following password policy operational attributes in user entries are only compared if both the supplier and consumer servers are participating in password policy:
   - **pwdChangedTime**
   - **pwdReset**
   - **ibm-pwdAccountLocked**
   - **pwdExpirationWarned**
   - **pwdGraceUseTime**
   - **pwdFailureTime**
   - **pwdAccountLockedTime**
   - **pwdHistory**

   Also, the following attributes for specifying individual and group policies in user and group entries are only compared if both the supplier and consumer servers are participating in password policy:
   - **ibm-pwdIndividualPolicyDn**
   - **ibm-pwdGroupPolicyDn**
10. If the supplier and consumer servers are participating in password policy, the **ldapdiff** utility needs to be run to synchronize the password policy entries in the **cn=ibmpolicies** subtrees on both servers, and then run to synchronize the entries in the LDBM and TDBM backends. There can be situations where the **ldapdiff** utility must be run for the entries in LDBM and TDBM subtrees first, and then run for the password policies entries. For example, if the consumer server has a password policy that no longer exists on the supplier server, but entries on the consumer server still reference that password policy, the password policy cannot be deleted until all references to it are removed. In this case, running **ldapdiff** for the LDBM and TDBM entries first removes those references, and then the **ldapdiff** utility can be run to synchronize the password policy entries.
11. The **ldapdiff** utility displays a message after it has finished comparing every 100th entry.

Diagnosis

Exit status is 0 if no errors occur. Errors result in a nonzero exit status and a diagnostic message being written to standard error.

ldapexop utility

Purpose

The **ldapexop** utility provides an interface to the **ldap_extended_operation()** API. The utility runs as a client operation while the server is active. The **ldapexop** utility performs the following extended operations:

- **acctstatus** - **Account status** extended operation
- **cascrepl** - **Cascading control replication** extended operation
- **controlqueue** - **Control replication queue** extended operation
- **controlrepl** - **Control replication** extended operation
- **controlreplerr** - **Control replication error log** extended operation
- **effectpwdpolicy** - **Effective password policy** extended operation

- **geteffectiveacl** - **GetEffectiveACL** extended operation
- **getusertype** - **User type** extended operation.
- **quiesce** - **Quiesce or unquiesce context** extended operation
- **repltopology** - **Replication topology** extended operation

Format

```
ldapexop [-d debugLevel] [-h host] [-p port] [-?] [-help] [-v]
   { [-m EXTERNAL -Z [-K keyFile] [-P keyFilePwd] [-N certificateLabel]]
     [ [ { [-D dn -w password] |
           [-m CRAM-MD5 -w password { [-D dn] | [-U userName [-D dn]] }] |
           [-m DIGEST-MD5 -U userName -w password [-D dn] [-G realmName]] |
           [-m GSSAPI]
         }
       ]
       [-Z [-K keyFile] [-P keyFilePwd] [-N certificateLabel]]
     ]
   } -op extendedOperation extOptions
```

Parameters

The options for the **ldapexop** utility are divided into two categories:

1. There are general options that specify how to connect to the LDAP server. These options must be specified before the **-op** option. See Table 29 for an explanation about the general options.
2. There are extended operation options that identify the extended operation to be performed. These options are specified after the general options and must begin with the **-op** option. See Extended operations options for an explanation about these options.

Option names can normally be specified in a reduced manner with as few as one or two of the initial characters specified.

All other command-line inputs result in a syntax error message, after the correct syntax is displayed. Except for the **–op** option, if the same option is specified multiple times, the last value specified is used. The **-op** option must be specified only once.

Table 29. ldapexop general options

Option	Description
-?	Display the usage.
-d *debugLevel*	Specify the level of the debug messages to be created. The debugLevel is specified in the same fashion as the debug level for the LDAP server. LDAP debug levels lists the specific debug levels. The default is no debug messages.
-D *dn*	Specify the DN to use to bind to the LDAP directory. The *dn* parameter is a string-represented DN. The default is a NULL string. If the **-m** option is equal to `DIGEST-MD5` or `CRAM-MD5`, this option is the authorization DN which is used for making access checks. This directive is optional when used in this manner.
-G *realmName*	Specify the realm name 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.

Table 29. ldapexop general options (continued)

Option	Description
-h *host*	Specify the host name or IP address on which the LDAP server is running. The default is the local host.
-help	Display the usage.
-K *keyFile*	Specify the name of the System SSL key database file, RACF key ring, or PKCS #11 token. If this option is not specified, this utility looks for the presence of the `SSL_KEYRING` environment variable with an associated name. If *keyFile* is specified as `*TOKEN*/`*NAME*, then System SSL uses the specified PKCS #11 token. Otherwise, System SSL uses a key database file or a RACF key ring. In this case, System SSL first assumes that *keyFile* is a key database file name and tries to locate the file. If *keyFile* is not a fully qualified z/OS UNIX System Services file name, the current directory is assumed to contain the key database file. The name cannot be a data set. If System SSL cannot locate the file, it then assumes that *keyFile* is a RACF key ring name. See "SSL/TLS information for LDAP utilities" on page 219 for information about System SSL key databases, RACF key rings, and PKCS #11 tokens. This parameter is ignored if **-Z** is not specified.
-m *mechanism*	Specify the bind method to use. Specify `GSSAPI` to indicate a Kerberos Version 5 bind 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 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. Unless you want to use the default certificate in the key database file, RACF key ring, or PKCS #11 token, use the **-N** option to specify the label of the certificate. 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. If **-m** is not specified, a simple bind is performed.
-N *certificateLabel*	Specify the label associated with the certificate in the key database file, RACF key ring, or PKCS #11 token. This parameter is ignored if **-Z** is not specified
-p *port*	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.

Table 29. ldapexop general options (continued)

Option	Description
-P *keyFilePwd*	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 file system file specification (for example, `file:///etc/ldap/sslstashfile`). The stash file must be a file and cannot be a data set. This parameter is ignored if **-Z** is not specified.
-U *userName*	Specify the user name for `CRAM-MD5` or `DIGEST-MD5` binds. The *userName* is a short name (uid) that is used to perform bind authentication. This option is required if the **-m** option is set to `DIGEST-MD5`.
-v	Use verbose mode, with many diagnostics written to standard output.
-w *password*	Specify the bind password for simple, `CRAM-MD5`, and `DIGEST-MD5` authentication. The default is a NULL string. Specify ? to prompt for the password.
-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** *keyFilePwd* option is required when the **-Z** option is specified and the key file specifies a file system key database file. Unless you want to use the default certificate in the key database file, RACF key ring, or PKCS #11 token, use the **-N** option to specify the label of the certificate.

Extended operations options: The **-op** *extendedOperation* option identifies the extended operation to be performed. The *extOptions* indicate the required and optional options for the specific extended operation. Each of the extended operations supported by the **ldapexop** utility is documented in detail in Appendix D, "Supported extended operations," on page 695. The supported extended operations and their options are:

acctstatus -d *userDn*

Perform the **Account status** extended operation. This extended operation returns the status of a user entry with a **userPassword** attribute value. The status returned is either opened, locked, or expired. See "Account status extended operation example" on page 390 for more information about the **Account status** extended operation.

-d *userDn*

Specifies the DN of the entry to be queried.

This example performs the **Account status** extended operation to query the status of the `cn=jon,o=acme,c=us` entry.

```
ldapexop -D adminDn -w adminPw -op acctstatus -d cn=jon,o=acme,c=us
```

cascrepl -action { quiesce | unquiesce | replnow | wait } -rc *contextDn* **[-timeout** *secs***]**

Perform the **Cascading control replication** extended operation. The requested action is applied to the specified server and also passed along to all replicas of the given context (subtree). If any of these are forwarding replicas or gateway servers, they pass the extended operation along to their replicas. The operation

cascades over the entire advanced replication topology. See "Cascading control replication" on page 696 for more information about the **Cascading control replication** extended operation.

-action { quiesce | unquiesce | replnow | wait }
Specifies the **Cascading control replication** extended operation action to be performed.

quiesce
Halts further updates, except by replication. Client updates under the specified context are restricted to LDAP administrators, if using the **Server Administration** control (OID 1.3.18.0.2.10.15), and any replication master DNs with authority under this context.

unquiesce
Resumes normal operation, client updates are accepted.

replnow
Replicates all queued changes to all replica servers as soon as possible, regardless of schedule. This operation is propagated to the consumer server of each replication agreement without waiting for all queued updates to be applied.

wait Replicates all queued changes to all replica servers as soon as possible, regardless of schedule. This operation is propagated to the consumer server of each replication agreement after all queued updates for that agreement are applied.

-rc *contextDn*
Specifies the DN of the advanced replication context (subtree).

-timeout *secs*
Specifies the number of seconds that the extended operation must successfully complete. If not present, or 0, the operation has an indefinite amount of time to complete.

This example performs the **Cascading control replication** extended operation to wait for all updates to be replicated to each consumer server in the `o=acme,c=us` replication context. If the extended operation does not successfully complete in 60 seconds, the operation ends with a return code of LDAP_TIMEOUT.

```
ldapexop –D adminDn –w adminPw -op cascrepl -action wait -rc "o=acme,c=us" -timeout 60
```

controlqueue -skip { all | *changeId* } -ra *agreementDn*
Perform the **Control replication queue** extended operation. This extended operation indicates which pending changes in the advanced replication queue are skipped and not replicated to the consumer server. See "Control replication queue" on page 703 for more information about the **Control replication queue** extended operation.

-skip {all | *changeId***}**

all Deletes (skips) all changes currently queued for replication for the specified replication agreement DN.

changeId
Deletes (skips) the change queued for replication with the change ID matching the specified number. The number must be in the range 1 - 4294967295. Change IDs can be determined by searching the *agreementDn* entry for the

ibm-replicationPendingChanges operational attribute. Only the next change to be replicated can be skipped in this manner.

-ra *agreementDn*
Specifies the DN of the advanced replication agreement.

This example performs the **Control replication queue** extended operation to delete (skip) all pending updates in the advanced replication queue for the replication agreement, `cn=server3,ibm-replicaSubentry=master1-id,ibm-replicaGroup=default,o=acme,c=us`. This extended operation prevents all changes in the replication queue from being replicated to all consumer servers.

```
ldapexop –D adminDn –w adminPw -op controlqueue -skip all
 -ra "cn=server3,ibm-replicaSubentry=master1-id,ibm-replicaGroup=default,o=acme,c=us"
```

controlrepl –action { suspend | resume | replnow } { -rc *contextDn* **| -ra** *agreementDn* **}**
Perform the **Control replication** extended operation. This extended operation indicates whether to suspend, resume, or immediately replicate entries in the advanced replication context (subtree) or agreement. See "Control replication" on page 700 for more information about the **Control replication** extended operation.

-action { suspend | resume | replnow }
Specifies the **Control replication** extended operation action to be performed.

suspend
Suspends advanced replication for the replication agreement or context (subtree). Any updates under the replication agreement or context are queued until the **Control replication** extended operation is performed to resume updates to consumer servers.

resume
Resumes advanced replication for the replication agreement or context (subtree).

replnow
Replicates immediately any outstanding updates for the replication agreement or context (subtree), regardless of schedule. **replnow** has no effect on a suspended replication agreement or context.

-rc *contextDn* **| -ra** *agreementDn*
Specifies the DN of the advanced replication context (subtree) or agreement the action is to be performed against. To perform the action against all agreements under a replication context, use **-rc** *contextDn* with the DN of the replication context. Alternatively, to perform this action against a single agreement, use **-ra** *agreementDn* with the DN of the replication agreement. A *contextDn* and an *agreementDn* cannot both be specified.

This example performs the **Control replication** extended operation to suspend advanced replication on the replication agreement, `cn=server3,ibm-replicaSubentry=master1-id,ibm-replicaGroup=default,o=acme,c=us`.

```
ldapexop –D adminDn –w adminPw -op controlrepl -action suspend
 -ra "cn=server3,ibm-replicaSubentry=master1-id,ibm-replicaGroup=default,o=acme,c=us"
```

controlreplerr -ra *agreementDn* **{ [-delete {** *failureId* **| all }] | [-retry {** *failureId* **| all }] | [-show** *failureId* **] }**
Perform the **Control replication error log** extended operation. This extended operation indicates whether to delete, retry, or show a failed advanced

replication update. Failing operations can be determined by searching the *agreementDn* entry for the **ibm-replicationFailedChanges** operational attribute which contains the failing change ID. See "Control replication error log" on page 702 for more information about the **Control replication error log** extended operation.

-ra *agreementDn*
Specifies the DN of the advanced replication agreement.

-delete { *failureId* **|** **all }**
Removes one or all failed replication updates.

failureId
Deletes only the failed update specified by the *failureId* for this agreement. The *failureId* must be in the range 1 - 4294967295.

all Deletes all the failed updates for this agreement.

-retry { *failureId* **|** **all }**
Reprocesses one or all failed replication updates.

failureId
Tries again only the failed update specified by the *failureId* for this agreement. The *failureId* must be in the range 1 - 4294967295.

all Tries again all the failed updates for this agreement.

-show *failureId*
Shows the failed update specified by the *failureId* for this agreement. The *failureId* must be in the range 1 - 4294967295.

This example performs the **Control replication error log** extended operation to remove all failures from the replication error log for the replication agreement, `cn=server3,ibm-replicaSubentry=master1-id,ibm-replicaGroup=default,o=acme,c=us`.

```
ldapexop –D adminDn –w adminPw -op controlreplerr -delete all
 -ra "cn=server3,ibm-replicaSubentry=master1-id,ibm-replicaGroup=default,o=acme,c=us"
```

effectpwdpolicy -d *entryDn*
Perform the **Effective password policy** extended operation. This extended operation returns the effective password policy entries and the attribute values of the effective password policy that the specified entry is subject to. See "Effective password policy" on page 705 for more information about the **Effective password policy** extended operation.

-d *entryDn*
Specifies the DN of the entry to be queried to obtain its effective password policy.

This example performs the **Effective password policy** extended operation to query the effective password policy of the `cn=jon,o=acme,c=us` entry.

```
ldapexop -D adminDn -w adminPw -op effectpwdpolicy -d cn=jon,o=acme,c=us
```

geteffectiveacl

Perform the **GetEffectiveACL** extended operation. This extended operation performs a search using the base DN, scope, and search filter to retrieve a list of entries. For each entry in the list, the extended operation then uses the optional bind and entry access information to determine the effective access control and returns the following:

- the entry DN to which access was requested

- the subject and all of its alternate DNs and group DNs for which access was calculated for
- the source attribute values (**aclEntry, aclPropagate, aclSource, entryOwner, ownerPropagate**, and **ownerSource**) in effect for the entry
- the applicable attribute values (**aclEntry** and **entryOwner**) used to form the effective permissions
- the calculated effective access class permissions
- the calculated effective attribute permissions

Note: Administrator permissions can be restricted by ACL filters that match an LDAP administrator DN. As a result, the calculated effective permissions of an LDAP administrator DN can have an affect on the results returned from the extended operation.

```
geteffectiveacl -filt searchFilter [-b baseDN]
  [-s {base | one | sub}] [-a {never | always | search | find}]
  [-z sizeLimit] [-l timeLimit]
  [-m {SIMPLE | CRAM-MD5 | DIGEST-MD5 | GSSAPI | EXTERNAL}]
  [-dn DN] [-u racfUserID] [-r principalRealm]
  [-ip bindIP] [-time accessTime] [-day accessDay] [-en]
```

Referral entries are not returned from the search operation. Therefore, access to referral entries is not calculated.

-filt[er] *searchFilter*
An IETF RFC 2254 (RFC 2254: *The String Representation of LDAP Search Filters*) compliant LDAP search filter that determines the entries to calculate effective permissions for. The *searchFilter* is required.

-b[ase] *baseDN*
The base DN used as a starting point for calculating effective permissions. If unspecified, *baseDN* is set to the empty string "", representing a null-based subtree starting point.

-s[cope] *scope*
The scope and *baseDN* determine the directory entries to which DN's permissions are calculated for, starting with *baseDN*. The scope must be **base, one**, or **sub**. Scope is only required if *baseDN* is unspecified. Otherwise, the default is **sub**.

-a *deref*
Specifies how alias dereferencing is done during the search operation. *deref* must be one of **never, always, search**, or **find** to specify that aliases are never dereferenced, always dereferenced, dereferenced when searching, or dereferenced only when locating the base object for the search. The default is **never**.

-z *sizeLimit*
Indicates the maximum number of entries to calculate effective permissions. A value of `0` indicates that there is no limit. The LDAP server can also provide a size limit on the number of entries returned in the **sizeLimit** configuration option. See "sizeLimit num-limit " on page 128 for more information about the LDAP server size limit. If this option is not specified or is set to `0`, the maximum number of entries returned is limited only by the LDAP server limit.

-l *timeLimit*
Indicates the maximum wait time (in seconds) for the search. A value of `0` indicates that there is no time limit on the search request. The LDAP server can also provide a limit on the search time in the **timeLimit** configuration

option. See “timeLimit num-seconds ” on page 135 for more information about the LDAP server time limit. If this option is not specified or is set to 0, the maximum number of seconds is limited only by the LDAP server limit.

-m[echanism] {SIMPLE | EXTERNAL | CRAM-MD5 | DIGEST-MD5 | GSSAPI}
The string representation of the mechanism the DN used to bind to the LDAP server. This is used to match the **ibm-filterBindMechanism** attribute in any ACL filters. The default value is **SIMPLE**.

-dn *DN*
Indicates the string representation of the DN to calculate effective permissions for. This is the bind DN for **SIMPLE, CRAM-MD5**, or **DIGEST-MD5** bind, and the subject DN for **EXTERNAL** bind. If **SIMPLE** bind is specified and this option is unspecified, the effective permissions are calculated for an anonymous bind. If **CRAM-MD5, DIGEST-MD5**, or **EXTERNAL** is used, this option is required. If **GSSAPI** bind is used, this option is ignored.

-u[serid] *racfUserID*
The string representation for the RACF user ID associated with the subject DN in the SSL certificate. This value is optional when **EXTERNAL** is specified. For all other bind mechanisms, this option is ignored.

-r[ealm] *principalRealm*
The SASL GSS API Kerberos bind source principal that is obtained from the GSS API client credentials, specified as *principal@REALM*. This value is required when **GSSAPI** is specified. For all other bind mechanisms, this option is ignored.

-ip *bindIP*
The string representation of the IPv4 or IPv6 address of the client used to connect to the LDAP server. If this value is unspecified, it defaults to the unspecified IPv4 address, *0.0.0.0*.

-time *accessTimeOfDay*
The string representation of the time of day the directory entry is accessed. The format is *hh:mm*, where *hh* ranges from 00 to 23, and *mm* ranges from 00 to 59. If this value is unspecified, it defaults to the server's current time of day.

-day *accessDayOfWeek*
The string representation of the day of week the directory entry is accessed. Valid values range from 0 to 6, where: *Sunday* = 0, *Monday* = 1, *Tuesday* = 2, *Wednesday* = 3, *Thursday* = 4, *Friday* = 5, *Saturday* = 6. If this value is unspecified, it defaults to the current day of week of the server.

-en[cryption]
When specified, it assumes the bind DN uses a secure LDAP connection (SSL). Otherwise, it assumes that the bind DN does not use a secure LDAP connection (SSL) with a non-clear cipher specification.

This example performs the **GetEffectiveACL** extended operation for each entry returned on the subtree search of the `dc=yourcompany,dc=com` subtree. The requested subtree search uses `dc=yourcompany,dc=com` as the *baseDN*, with a filter of "objectclass=*", a search size limit of `100`, a search time limit of `10` seconds, and no alias dereferencing. Based on these returned search entries, the **GetEffectiveACL** extended operation calculates the effective ACLs for user `cn=Joe Shmoe,ou=users,dc=yourcompany,dc=com` when a simple bind is done

from IP address 129.176.132.92 at 18:30 on a Saturday over a secure SSL connection. See "Querying effective permissions" on page 452 for the output of this example.

```
ldapexop -D adminDn -w adminPw -op geteffectiveacl -filter "objectclass=*"
-base "dc=yourcompany,dc=com" -s sub -a never -z 100 -l 10
-dn "cn=Joe Shmoe,ou=users,dc=yourcompany,dc=com" -ip 129.176.132.92
-time 18:30 -day 6 -mech SIMPLE –encrypt
```

getusertype

Perform the **User type** extended operation. This extended operation provides the authenticated user with their user type and administrative roles. See "User type" on page 719 for more information about the **User Type** extended operation.

This example performs the **User type** extended operation to determine the user type and roles for user cn=jon,o=ibm,c=us:

```
ldapexop -D cn=jon,o=ibm,c=us -w secret -op getusertype
```

quiesce -rc *contextDn* **[-end]**

Perform the **Quiesce or unquiesce context** extended operation. When an advanced replication context is quiesced, update operations are only accepted from certain users. See "Quiesce or unquiesce context" on page 711 for more information about the **Quiesce or unquiesce context** extended operation.

-rc *contextDN*
: Specifies the DN of the advanced replication context (subtree) to be quiesced or unquiesced.

-end
: Unquiesces the advanced replication context (subtree). If not specified, the default is to quiesce the replication context.

This example performs the **Quiesce or unquiesce context** extended operation to quiesce the replication context o=acme,c=us. After this extended operation is performed, updates to entries within the replication context are only allowed from certain users.

```
ldapexop –D adminDn –w adminPw -op quiesce -rc "o=acme,c=us"
```

repltopology -rc *contextDn* **[-timeout** *secs***] [-ra** *agreementDn***]**

Perform the **Replication topology** extended operation. This extended operation synchronizes all replication topology related entries under the specified context DN. See "Replication topology" on page 714 for more information about the **Replication topology** extended operation.

-rc *contextDN*
: Specifies the DN of the advanced replication context (subtree) on the supplier server for which advanced replication topology related entries are synchronized. The extended operation is cascaded through all forwarding and gateway servers.

-timeout *secs*
: Specifies the number of seconds that the extended operation must successfully complete. If not present, or 0, the operation has an indefinite amount of time to complete.

-ra *agreementDn*
: Specifies the DN of the advanced replication agreement and allows the operation to be restricted to only one server and its consumer.

This example performs the **Replication topology** extended operation to synchronize replication topology entries within the o=acme,c=us replication context with the consumer defined in the replication agreement cn=server3,ibm-replicaSubentry=master1-id,ibm-

replicaGroup=default,o=acme,c=us. This extended operation does not cascade, it only modifies the consumer defined in the replication agreement before returning.

```
ldapexop –D adminDn –w adminPw -op repltopology -rc "o=acme,c=us"
 -ra "cn=server3,ibm-replicaSubentry=master1-id,ibm-replicaGroup=default,o=acme,c=us"
```

Usage

1. You must be bound as an LDAP administrator with the appropriate authority to successfully perform all extended operations (other than the **User type** extended operation) against the z/OS LDAP server. See Table 16 on page 168 for information about which extended operations are allowed to be performed by administrative group members.
2. The **ldapexop** utility sends the **PasswordPolicy** control as a non-critical control when the user attempts to authenticate to the targeted LDAP server. If the bound user is subject to password policy on the LDAP server, the **ldapexop** utility parses and displays the warning and error messages from the **PasswordPolicy** control response.
3. The LDAP_DEBUG environment variable can be used to set the debug level. For more information about specifying the debug level using keywords, decimal, hexadecimal, and plus and minus syntax, see Table 18 on page 173 for a list of specific debug levels.
4. You can specify an LDAP URL for host on the **-h** parameter. See **ldap_init()** in *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information.
5. The password prompt (`-w ?`) is not supported when running from TSO or batch. In these environments, the password value must be specified on the **-w** option.

   The **getpass()** routine used to prompt for the password returns at most PASS_MAX number of characters, truncating any additional characters. See the description of **getpass()** in *z/OS XL C/C++ Runtime Library Reference* for more information. If the length of the specified password is greater than PASS_MAX, the password value must be specified on the **-w** option.

Diagnosis

Exit status is 0 if no errors occur. Errors result in a nonzero exit status and a diagnostic message being written to standard error.

See Appendix D, “Supported extended operations,” on page 695 for additional diagnostic information about each of the extended operations.

Chapter 13. Globalization support

This topic contains information about translated messages and UTF-8 support.

Translated messages

The **LANG** and **NLSPATH** environment variables are set for the z/OS LDAP server and utility programs in the LDAP environment variables file. The default name and location for this file is:

```
/etc/ldap/ds.envvars
```

A sample **ds.envvars** file 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
LANG=En_US.IBM-1047
```

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 runs the LDAP directory utilities.

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 30 shows the mapping between Unicode and UTF-8. See RFC 2279: *UTF-8, a transformation format of ISO 10646* and RFC 2253: *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names* for more information about UTF-8.

Table 30. 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 LDAP server supports UTF-8 data exchange as part of its Version 3 protocol support.

Note: For UTF-8 data stored in a LDAP server's TDBM and GDBM (when DB2-based) 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 14. 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* that are required to be present in the entry and 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 directory 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.

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
```

The order of the component attribute=value pairs is important. The DN contains one component for each level of the directory hierarchy. LDAP directory DNs begin with the most specific attribute (typically 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 that make 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. The TDBM, LDBM, and CDBM backends add 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, therefore, 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 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
```

Any attribute can be used to make up an RDN except:

- Attributes with binary syntax.

  Note: The **userPassword** attribute is binary, therefore, it cannot appear in an RDN.
- Attributes that are marked NO-USER-MODIFICATION in the schema, because these attributes cannot be added to an entry by a user.
- The **aclEntry, aclPropagate, entryOwner**, and **ownerPropagate** attributes.

Distinguished name syntax

The Distinguished Name (DN) syntax supported by this server is based on RFC 2253: *Lightweight Directory Access Protocol (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. A plus sign (+) is used to separate attribute=value pairs in an RDN.

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 number 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. Number signs (#) and space characters that do not occur at the beginning of a string can also be escaped, but this is not required.

This notation is 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 in RFC 2247: *Using Domains in LDAP/X.500 Distinguished Names* 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 distinguished 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 contains three required attributes plus a suffix:

racfuserid+racfgroupid
Specifies the user and the group.

profiletype
Specifies **connect**.

suffix Specifies the SDBM suffix.

A RACF-style DN for a general resource profile contains two required attributes plus a suffix:

profilename
Specifies the name of the resource profile. The case of the name is important if the class containing the resource profile supports mixed case names.

profiletype
The name of the class containing the resource profile.

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 DN format and a sample DN for a user:

```
racfid=userid,profiletype=user,suffix
racfid=ID1,profiletype=user,cn=myRACF,c=US
```

Following is 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
```

Following is the DN format and a sample DN for a resource profile:

```
profilename=resourcename,profiletype=classname,suffix
profilename=ABC.KEN,profiletype=FACILITY,cn=myRACF,c=US
```

Chapter 15. LDAP directory schema

The LDAP Version 3 (V3) protocol, as defined in RFC 2252: *Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions* and RFC 2256: *A Summary of the X.500 (96) User Schema for use with LDAPv3*, describes schema publication and update. Schema publication enables you 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 implements both schema publication and update. The schema is stored as an entry in the database and search (publication) and modify (update) operations might be performed on this entry. The distinguished name of the schema entry is `cn=schema`.

  The **schemaPath** option in the LDAP server configuration file defines the location where the LDAP server saves the schema entry. The default is **/var/ldap/schema**. This directory is backed up as part of the normal system backup procedure since the loss of the schema directory invalidates all existing directory entries. If there are multiple LDAP servers running in single-server on the system, a unique schema directory must be specified in the **schemaPath** configuration option for each LDAP server. If there are multiple LDAP servers running in multi-server mode in the same sysplex group on the system, the schema directory specified in the **schemaPath** configuration option must be the same in each LDAP server's configuration file and must exist within a shared z/OS UNIX System Services file system. See “Determining operational mode” on page 146 for more information.
- When the z/OS LDAP server is first started, the server supplies an initial schema. This initial schema is sufficient for usage of the SDBM (without RACF custom fields), CDBM (with configuration-related entries), and GDBM backends, but must be updated for usage of LDBM, TDBM, SDBM with RACF custom fields, and CDBM with user-defined entries. The initial schema elements cannot be deleted and can only be modified in limited ways. See Appendix A, “Initial LDAP server schema,” on page 633 for the contents of the initial schema.
- Access to the schema entry is controlled by an access control list (ACL), even if the LDAP server is in maintenance mode. All requests to access the schema entry, except those from an LDAP root or schema administrator, are subject to ACL checking. In particular for a basic replication replica server, requests from the **masterServerDN** or **peerServerDN** are subject to access control. The default ACL allows all users to display the schema, but only an LDAP root or schema administrator can update the schema. This ACL can be modified. See Chapter 24, “Using access control,” on page 433 for more information.

Setting up the schema for LDBM, TDBM, and CDBM

The LDAP server is shipped with two predefined schema files representing schema definitions that the user might want to load into the LDAP server schema when using LDBM, TDBM, or CDBM. These files are **schema.user.ldif** and **schema.IBM.ldif** and are in the **/usr/lpp/ldap/etc** directory. These files contain schema definitions that 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.

If you are creating a new z/OS IBM TDS, determine which of these schema files, if any, could be used to represent the data to be stored in the LDBM, TDBM, or CDBM directory, or locate or create other schema files to use. The schema definitions in **schema.user.ldif** are required to be loaded before loading the **schema.IBM.ldif**. If additional schema definitions are required from other products or customized applications, load them after **schema.user.ldif** and **schema.IBM.ldif** files are loaded.

Use the **ldapmodify** utility to load the schema or change the existing **cn=schema** entry on the server. For example, the **ldapmodify** utility commands used to load the **schema.user.ldif** and **schema.IBM.ldif** schema files would be:

```
ldapmodify -h ldaphost -p ldapport -D adminDN -w passwd -f /usr/lpp/ldap/etc/schema.user.ldif
ldapmodify -h ldaphost -p ldapport -D adminDN -w passwd -f /usr/lpp/ldap/etc/schema.IBM.ldif
```

See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information about **ldapmodify**.

Note:

1. Check that **schemaReplaceByValue off** is not specified in the global section of the LDAP server configuration file, or send the **IBMSchemaReplaceByValueControl** control with a value of **TRUE** on the modify request. This control can be sent by specifying the **-u** option on the **ldapmodify** utility. See "schemaReplaceByValue {on | off}" on page 123 for more information about the **schemaReplaceByValue** configuration option and "IBMSchemaReplaceByValueControl" on page 682 for more information about the **IBMSchemaReplaceByValueControl** control.
2. When the LDAP schema is modified using the **schema.user.ldif** and **schema.IBM.ldif** files, each attribute and object class definition in the file replaces the existing definition in the schema. Any changes previously made in the schema to these attributes and object classes must be made again. This includes any changes that are allowed to attributes and object classes in the initial LDAP schema.

If you are migrating an existing z/OS IBM TDS from an earlier release, the schema definitions that are supplied in **schema.user.ldif** or **schema.IBM.ldif** do not need to be reapplied. If the initial schema has changed from the previous release, the LDAP server automatically updates the schema with any new attribute types and object classes that have been added.

Schema introduction

Entries in the directory are made up of attributes that 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=`*value* pairs that identify what type of information the entry contains. The object classes that are associated with the entry determine the set of attributes that must or might be present in the entry.

The z/OS LDAP server has a single schema for the entire server. This schema is stored as an entry whose distinguished name is `cn=schema`. Following is a portion of the schema entry.

```
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 )
...
```

Figure 8. 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, `cn=schema`. Therefore, a search operation requesting the **subschemasubentry** for an entry always returns:

```
subschemasubentry=cn=schema
```

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 might 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.

Note: Non-numeric object identifiers, for example `myattr-oid`, can be used instead of numeric object identifiers. However, non-numeric identifiers are not supported in the z/OS Integrated Security Services LDAP server on earlier releases. Do not use non-numeric identifiers if you intend to share a TDBM database with a z/OS Integrated Security Services LDAP server on earlier releases.

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 34 on page 288, Table 35 on page 289, and Table 36 on page 290. 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, number of values, 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 fails. The syntaxes supported by the z/OS LDAP server are shown in Table 34 on page 288 and Table 35 on page 289.

Matching rules might be specified for **EQUALITY**, **ORDERING**, and **SUBSTR** (substring matching). The matching rule determines how comparisons between values are 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 performs evaluations to the best of its ability, but the results might 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, ORDERING**, and **SUBSTR** if those matching rules are not specified.

Table 31. Syntax and default EQUALITY, ORDERING, and SUBSTR matching rules

Syntax	EQUALITY	ORDERING	SUBSTR
LDAP syntaxes and matching rules marked with a * are only supported when the LDAP server is running at server compatibility level 6 or higher.			
Attribute Type Description	objectIdentifierFirstComponentMatch	-	-
Binary	-	-	-
Bit String*	bitStringMatch*	-	-
Boolean	booleanMatch	-	-
Certificate*	certificateMatch*	-	-
Certificate List*	-	-	-
Certificate Pair*	-	-	-
Country String*	caseIgnoreMatch	caseIgnoreOrderingMatch	caseIgnoreSubstringsMatch
Delivery Method*	caseIgnoreMatch	caseIgnoreOrderingMatch	caseIgnoreSubstringsMatch
Directory String	caseIgnoreMatch	caseIgnoreOrderingMatch	caseIgnoreSubstringsMatch
Distinguished Name	distinguishedNameMatch	distinguishedNameOrderingMatch	-
DIT Content Rule Description	objectIdentifierFirstComponentMatch	-	-
DIT Structure Rule Description	integerFirstComponentMatch	-	-
Enhanced Guide*	caseIgnoreMatch	caseIgnoreOrderingMatch	caseIgnoreSubstringsMatch
Facsimile Telephone Number*	telephoneNumberMatch	-	telephoneNumberSubstringsMatch
Fax*	-	-	-
Generalized Time	generalizedTimeMatch	generalizedTimeOrderingMatch	-

Table 31. Syntax and default EQUALITY, ORDERING, and SUBSTR matching rules (continued)

Syntax	EQUALITY	ORDERING	SUBSTR
LDAP syntaxes and matching rules marked with a * are only supported when the LDAP server is running at server compatibility level 6 or higher.			
Guide*	caseIgnoreMatch	caseIgnoreOrderingMatch	caseIgnoreSubstringsMatch
IA5 String	caseIgnoreIA5Match	caseIgnoreOrderingMatch	caseIgnoreIA5SubstringsMatch*
IBM Attribute Type Description	objectIdentifierFirstComponentMatch	-	-
IBM Entry UUID	IBM-EntryUUIDMatch	-	-
Integer	integerMatch	integerOrderingMatch*	-
JPEG*	-	-	-
LDAP Syntax Description	objectIdentifierFirstComponentMatch	-	-
Matching Rule Description	objectIdentifierFirstComponentMatch	-	-
Matching Rule Use Description	objectIdentifierFirstComponentMatch	-	-
MHS OR Address*	caseIgnoreMatch	caseIgnoreOrderingMatch	caseIgnoreSubstringsMatch
Name And Optional UID*	uniqueMemberMatch*	distinguishedNameOrderingMatch	-
Name Form Description	objectIdentifierFirstComponentMatch	-	-
Numeric String*	numericStringMatch*	numericStringOrderingMatch*	numericStringSubstringsMatch*
Object Class Description	objectIdentifierFirstComponentMatch	-	-
Object Identifier	objectIdentifierMatch	-	-
Octet String	octetStringMatch	octetStringOrderingMatch*	-
Other Mailbox*	caseIgnoreMatch	caseIgnoreOrderingMatch	caseIgnoreSubstringsMatch
Postal Address*	caseIgnoreListMatch*	caseIgnoreOrderingMatch	caseIgnoreListSubstringsMatch*
Presentation Address*	presentationAddressMatch*	caseIgnoreOrderingMatch	caseIgnoreSubstringsMatch
Printable String*	caseIgnoreMatch	caseIgnoreOrderingMatch	caseIgnoreSubstringsMatch
Protocol Information*	protocolInformationMatch*	caseIgnoreOrderingMatch	caseIgnoreSubstringsMatch
Substring Assertion	-	-	-
Supported Algorithm*	caseIgnoreMatch	caseIgnoreOrderingMatch	caseIgnoreSubstringsMatch
Telephone Number	telephoneNumberMatch	-	telephoneNumberSubstringsMatch
Teletex Terminal Identifier*	caseIgnoreMatch	caseIgnoreOrderingMatch	caseIgnoreSubstringsMatch
Telex Number*	telephoneNumberMatch	-	telephoneNumberSubstringsMatch
UTC Time	utcTimeMatch	generalizedTimeOrderingMatch	-

The z/OS LDAP server also verifies that the matching rules specified for **EQUALITY**, **ORDERING**, and **SUBSTR** are consistent with the specified **SYNTAX**. Table 32 shows acceptable values **EQUALITY**, **ORDERING**, and **SUBSTR**.

Table 32. Syntax and acceptable matching rules (EQUALITY, ORDERING, and SUBSTR)

Syntax	EQUALITY	ORDERING	SUBSTR
LDAP syntaxes and matching rules marked with a * are only supported when the LDAP server is running at server compatibility level 6 or higher.			
Attribute Type Description	objectIdentifierFirstComponentMatch	-	-
Binary	-	-	-
Bit String*	bitStringMatch*	-	-
Boolean	booleanMatch caseIgnoreMatch caseExactMatch	-	-
Certificate*	certificateMatch* certificateExactMatch*	-	-
Certificate List*	-	-	-

Table 32. Syntax and acceptable matching rules (EQUALITY, ORDERING, and SUBSTR) (continued)

Syntax	EQUALITY	ORDERING	SUBSTR
LDAP syntaxes and matching rules marked with a * are only supported when the LDAP server is running at server compatibility level 6 or higher.			
Certificate Pair*	-	-	-
Country String*	caseIgnoreMatch	caseIgnoreOrderingMatch	caseIgnoreSubstringsMatch
Delivery Method*	caseIgnoreMatch caseIgnoreIA5Match caseIgnoreListMatch* presentationAddressMatch* protocolInformationMatch* caseExactMatch caseExactIA5Match	caseIgnoreOrderingMatch caseExactOrderingMatch	caseIgnoreSubstringsMatch caseIgnoreIA5SubstringsMatch* caseIgnoreListSubstringsMatch* caseExactSubstringsMatch
Directory String	caseIgnoreMatch caseExactMatch	caseIgnoreOrderingMatch caseExactOrderingMatch	caseIgnoreSubstringsMatch caseExactSubstringsMatch
Distinguished Name	distinguishedNameMatch uniqueMemberMatch*	distinguishedNameOrdering Match	-
DIT Content Rule Description	objectIdentifierFirstComponentMatch	-	-
DIT Structure Rule Description	integerFirstComponentMatch	-	-
Enhanced Guide*	caseIgnoreMatch caseIgnoreIA5Match caseIgnoreListMatch* presentationAddressMatch* protocolInformationMatch* caseExactMatch caseExactIA5Match	caseIgnoreOrderingMatch caseExactOrderingMatch	caseIgnoreSubstringsMatch caseIgnoreIA5SubstringsMatch* caseIgnoreListSubstringsMatch* caseExactSubstringsMatch
Facsimile Telephone Number*	telephoneNumberMatch	-	telephoneNumberSubstringsMatch
Fax*	-	-	-
Generalized Time	generalizedTimeMatch	generalizedTimeOrdering Match	-
Guide*	caseIgnoreMatch caseIgnoreIA5Match caseIgnoreListMatch* presentationAddressMatch* protocolInformationMatch* caseExactMatch caseExactIA5Match	caseIgnoreOrderingMatch caseExactOrderingMatch	caseIgnoreSubstringsMatch caseIgnoreIA5SubstringsMatch* caseIgnoreListSubstringsMatch* caseExactSubstringsMatch
IA5 String	caseIgnoreMatch caseIgnoreIA5Match caseExactMatch caseExactIA5Match	caseIgnoreOrderingMatch caseExactOrderingMatch	caseIgnoreSubstringsMatch caseIgnoreIA5SubstringsMatch* caseExactSubstringsMatch
IBM Attribute Type Description	objectIdentifierFirstComponent Match	-	-
IBM Entry UUID	IBM-EntryUUIDMatch	-	-
Integer	integerMatch integerFirstComponentMatch	integerOrderingMatch*	-
JPEG*	-	-	-
LDAP Syntax Description	objectIdentifierFirstComponentMatch	-	-
Matching Rule Description	objectIdentifierFirstComponentMatch	-	-
Matching Rule Use Description	objectIdentifierFirstComponentMatch	-	-

Table 32. Syntax and acceptable matching rules (EQUALITY, ORDERING, and SUBSTR) (continued)

Syntax	EQUALITY	ORDERING	SUBSTR
LDAP syntaxes and matching rules marked with a * are only supported when the LDAP server is running at server compatibility level 6 or higher.			
MHS OR Address*	caseIgnoreMatch caseIgnoreIA5Match caseIgnoreListMatch* presentationAddressMatch* protocolInformationMatch* caseExactMatch caseExactIA5Match	caseIgnoreOrderingMatch caseExactOrderingMatch	caseIgnoreSubstringsMatch caseIgnoreIA5SubstringsMatch* caseIgnoreListSubstringsMatch* caseExactSubstringsMatch
Name And Optional UID*	distinguishedNameMatch uniqueMemberMatch*	distinguishedNameOrderingMatch	-
Name Form Description	objectIdentifierFirstComponentMatch	-	-
Numeric String*	numericStringMatch*	numericStringOrderingMatch*	numericStringSubstringsMatch*
Object Class Description	objectIdentifierFirstComponentMatch	-	-
Object Identifier	objectIdentifierMatch objectIdentifierFirstComponentMatch	-	-
Octet String	octetStringMatch	octetStringOrderingMatch*	-
Other Mailbox*	caseIgnoreMatch caseIgnoreIA5Match caseIgnoreListMatch* presentationAddressMatch* protocolInformationMatch* caseExactMatch caseExactIA5Match	caseIgnoreOrderingMatch caseExactOrderingMatch	caseIgnoreSubstringsMatch caseIgnoreIA5SubstringsMatch* caseIgnoreListSubstringsMatch* caseExactSubstringsMatch
Postal Address*	caseIgnoreMatch caseIgnoreIA5Match caseIgnoreListMatch* presentationAddressMatch* protocolInformationMatch* caseExactMatch caseExactIA5Match	caseIgnoreOrderingMatch caseExactOrderingMatch	caseIgnoreSubstringsMatch caseIgnoreIA5SubstringsMatch* caseIgnoreListSubstringsMatch* caseExactSubstringsMatch
Presentation Address*	caseIgnoreMatch caseIgnoreIA5Match caseIgnoreListMatch* presentationAddressMatch* protocolInformationMatch* caseExactMatch caseExactIA5Match	caseIgnoreOrderingMatch caseExactOrderingMatch	caseIgnoreSubstringsMatch caseIgnoreIA5SubstringsMatch* caseIgnoreListSubstringsMatch* caseExactSubstringsMatch
Printable String*	caseIgnoreMatch caseIgnoreIA5Match caseIgnoreListMatch* presentationAddressMatch* protocolInformationMatch* caseExactMatch caseExactIA5Match	caseIgnoreOrderingMatch caseExactOrderingMatch	caseIgnoreSubstringsMatch caseIgnoreIA5SubstringsMatch* caseIgnoreListSubstringsMatch* caseExactSubstringsMatch
Protocol Information*	caseIgnoreMatch caseIgnoreIA5Match caseIgnoreListMatch* presentationAddressMatch* protocolInformationMatch* caseExactMatch caseExactIA5Match	caseIgnoreOrderingMatch caseExactOrderingMatch	caseIgnoreSubstringsMatch caseIgnoreIA5SubstringsMatch* caseIgnoreListSubstringsMatch* caseExactSubstringsMatch
Substring Assertion	-	-	-

Table 32. Syntax and acceptable matching rules (EQUALITY, ORDERING, and SUBSTR) (continued)

Syntax	EQUALITY	ORDERING	SUBSTR
LDAP syntaxes and matching rules marked with a * are only supported when the LDAP server is running at server compatibility level 6 or higher.			
Supported Algorithm*	caseIgnoreMatch caseIgnoreIA5Match caseIgnoreListMatch* presentationAddressMatch* protocolInformationMatch* caseExactMatch caseExactIA5Match	caseIgnoreOrderingMatch caseExactOrderingMatch	caseIgnoreSubstringsMatch caseIgnoreIA5SubstringsMatch* caseIgnoreListSubstringsMatch* caseExactSubstringsMatch
Telephone Number	telephoneNumberMatch	-	telephoneNumberSubstringsMatch
Teletex Terminal Identifier*	caseIgnoreMatch caseIgnoreIA5Match caseIgnoreListMatch* presentationAddressMatch* protocolInformationMatch* caseExactMatch caseExactIA5Match	caseIgnoreOrderingMatch caseExactOrderingMatch	caseIgnoreSubstringsMatch caseIgnoreIA5SubstringsMatch* caseIgnoreListSubstringsMatch* caseExactSubstringsMatch
Telex Number*	telephoneNumberMatch	-	telephoneNumberSubstringsMatch
UTC Time	utcTimeMatch generalizedTimeMatch	generalizedTimeOrderingMatch	-

The syntax or matching rule values might 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 might 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 are used to determine these values. The **SYNTAX** must be specified on the attribute type or through inheritance.

The number of values that might 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 might 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 that contain data values of an attribute type which has been made obsolete fails unless all data values for all obsolete attribute types are removed during the modification. Searches specifying the obsolete attribute type returns the entries containing the attribute type. If an obsolete attribute type is referred to in a superior hierarchy, the inherited values 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 that has an operational USAGE value, is called an operational attribute while all other attribute types are also known as non-operational attribute types. 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. If a plus sign, '**+**', is specified in the list of attributes to be returned, then all operational attributes other than **ibm-allMembers, ibm-allGroups, ibm-entryCheckSum, ibm-entryCheckSumOp**, and **hasSubordinates** are returned on the search response, if the user is authorized to read those operational attributes. Also, operational attribute types do not have to belong to an object class. The default for **USAGE** is **userApplications**. By default, non-operational attribute types are returned on a search response if the user is authorized to read those attributes. If an asterisk, '*****', is specified in the list of attributes to be returned, then all non-operational attributes are returned. Specifying both '*****' and '**+**' returns non-operational and operational attribute types that the user is authorized to read.

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. In general, **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. Attribute types which are **NO-USER-MODIFICATION** can be modified during replication processing and when the LDAP server is in maintenance mode. See “Basic replication maintenance mode” on page 475 and “Advanced replication maintenance mode” on page 536 for more information.

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** and the **RACFFIELD** of the associated attribute type.

  ACCESS-CLASS specifies the level of access users have to data values of this attribute type. The levels that might be specified for user-defined attribute types are **normal**, **sensitive**, and **critical**. The **system** and **restricted** keywords are for LDAP server use and are specified for some of the attribute types controlled by the server. See “Attribute access classes” on page 437 for the definition of access classes.

  RACFFIELD specifies the information needed to associate this attribute type with a RACF custom field. The presence of **RACFFIELD** indicates that this attribute type is used to represent a RACF custom field in a user or group profile.

Note: Other LDAP servers from IBM use the **DBNAME** and **LENGTH** characteristics to specify additional information for their implementations. These might 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 might 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 might be created with multiple levels of inheritance.

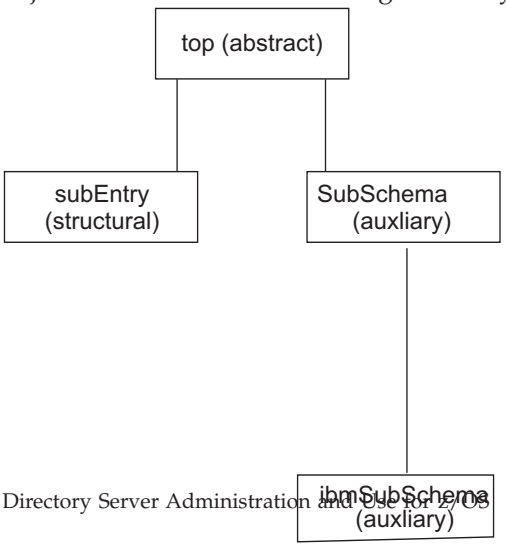
  Each object class is defined as one of three types: **STRUCTURAL**, **ABSTRACT**, or **AUXILIARY**. The type can be specified when the object class is defined. If the type is not specified, it defaults to **STRUCTURAL**.

  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 cannot be changed.** Once an entry is defined in the directory, it must be deleted and re-created 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 8 on page 277. The schema entry specifies **top**, **subEntry**, **subSchema**, and **ibmSubschema** as object classes. The object classes form the following hierarchy:



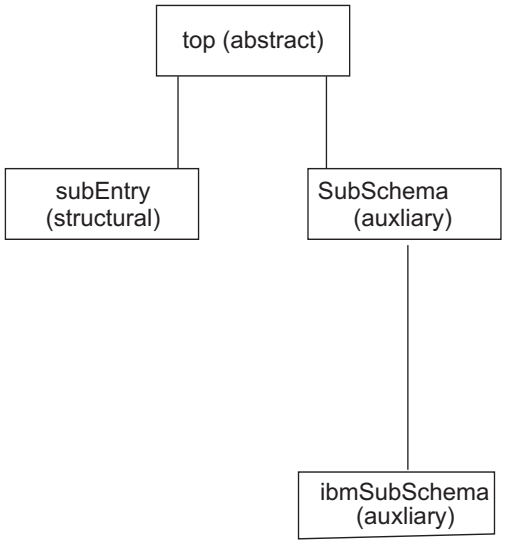


Figure 9. Object class hierarchy example

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 fails.

- **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 34 on page 288 and Table 35 on page 289 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. Some syntaxes are only supported when the LDAP server is running at server compatibility level 6 or higher. See "LDAP schema attributes" on page 287 for more information.

- **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 might 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 36 on page 290 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. Some matching rules are only supported when the LDAP server is running at server compatibility level 6 or higher. See "LDAP schema attributes" on page 287 for more information.

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 33. The terms shown in this table are used in the schema attribute definitions in the next section.

Table 33. 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 must handle this during data input.
numericoid	A dotted decimal string. Example: `2.5.13.72` **Note:** A non-numeric object identifier, for example `myattr-oid`, can be used instead of a numeric object identifier. However, non-numeric identifiers are not supported in the z/OS Integrated Security Services LDAP server on earlier releases. Do not use non-numeric identifiers if you intend to share a TDBM database with a z/OS Integrated Security Services LDAP server on earlier releases.

oid	A single object identifier. This might 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 33. 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 quotation marks (').
	
	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 might 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

Below contains information about the five attributes used to define an LDAP schema. For these schema attributes, the *numericoid* must be the first item in the definition. All other keywords and values might be in any order.

Note: Some syntaxes and matching rules are only supported when the LDAP server is running at server compatibility level 6 or higher. These syntaxes and matching rules are indicated by a * in the tables below. See “serverCompatLevel {3 | 4 | 5 | 6 | 7| 8}” on page 125 for more information about setting the server compatibility level. When the server compatibility level is less than 6, these syntaxes and matching rules are not displayed as part of the schema, attributes using these syntaxes or matching rules cannot be created, and the LDAP server does not start if any attributes using these syntaxes or matching rules are already in the schema. Existing attributes using these syntaxes or matching rules must be deleted or modified to not use them while the server compatibility level is 6 or higher before the server can be restarted with a compatibility level less than 6.

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 34, would be used when defining attribute types that are used for directory data.

Table 34. Supported LDAP syntaxes - general use

Numeric object identifier	Description	Valid values
LDAP syntaxes marked with a * are only supported when the LDAP server is running at server compatibility level 6 or higher.		
1.3.6.1.4.1.1466.115.121.1.5	Binary	Binary data
1.3.6.1.4.1.1466.115.121.1.6	Bit String*	Bit data format (for example '0110'B)
1.3.6.1.4.1.1466.115.121.1.7	Boolean	TRUE, FALSE
1.3.6.1.4.1.1466.115.121.1.8	Certificate*	Binary data (no format checking)
1.3.6.1.4.1.1466.115.121.1.9	Certificate List*	Binary data (no format checking)
1.3.6.1.4.1.1466.115.121.1.10	Certificate Pair*	Binary data (no format checking)
1.3.6.1.4.1.1466.115.121.1.11	Country String*	2 printable characters (alphabetic, digits, **'**, **(**, **)**, **+**, **,**, **-**, **.**, **/**, **:**, **?**, space)
1.3.6.1.4.1.1466.115.121.1.12	Distinguished Name	Sequence of attribute type and value pairs
1.3.6.1.4.1.1466.115.121.1.14	Delivery Method*	UTF-8 characters
1.3.6.1.4.1.1466.115.121.1.15	Directory String	UTF-8 characters
1.3.6.1.4.1.1466.115.121.1.21	Enhanced Guide*	UTF-8 characters (no format checking)
1.3.6.1.4.1.1466.115.121.1.22	Facsimile Telephone Number*	Printable string (alphabetic, digits, **'**, **(**, **)**, **+**, **,**, **-**, **.**, **/**, **:**, **?**, space) and **$** (no format checking)
1.3.6.1.4.1.1466.115.121.1.23	Fax*	Binary data (no format checking)
1.3.6.1.4.1.1466.115.121.1.27 **Note:** The effective time zone for the LDAP server is assumed when calculating Coordinated Universal Time from local time.	Integer	*yyyymmddhhmmss.ffffff* (local time) *yyyymmddhhmmss.ffffff*Z (Coordinated Universal Time) *yyyymmddhhmmss.ffffff-hhmm* (Time zone west) *yyyymmddhhmmss.ffffff+hhmm* (Time zone east) The seconds (*ss*) and microseconds (*ffffff*) can be omitted and defaults to 0.
1.3.6.1.4.1.1466.115.121.1.25	Guide*	UTF-8 characters (no format checking)
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
1.3.6.1.4.1.1466.115.121.1.28	JPEG*	Binary data (no format checking)
1.3.6.1.4.1.1466.115.121.1.33	MHS OR Address*	UTF-8 characters (no format checking)
1.3.6.1.4.1.1466.115.121.1.34	Name And Optional UID*	Sequence of attribute type and value pairs

Table 34. Supported LDAP syntaxes - general use (continued)

Numeric object identifier	Description	Valid values
LDAP syntaxes marked with a * are only supported when the LDAP server is running at server compatibility level 6 or higher.		
1.3.6.1.4.1.1466.115.121.1.36	Numeric String*	List of space-separated numbers
1.3.6.1.4.1.1466.115.121.1.38	Object Identifier	Name or numeric object identifier
1.3.6.1.4.1.1466.115.121.1.39	Other Mailbox*	UTF-8 characters (no format checking)
1.3.6.1.4.1.1466.115.121.1.40	Octet String	Octet data
1.3.6.1.4.1.1466.115.121.1.41	Postal Address*	UTF-8 characters (no format checking)
1.3.6.1.4.1.1466.115.121.1.42	Protocol Information*	UTF-8 characters (no format checking)
1.3.6.1.4.1.1466.115.121.1.43	Presentation Address*	UTF-8 characters (no format checking)
1.3.6.1.4.1.1466.115.121.1.44	Printable String*	Printable string (alphabetic, digits, **'**, **(**, **)**, **+**, **,**, **-**, **.**, **/**, **:**, **?**, space)
1.3.6.1.4.1.1466.115.121.1.49	Supported Algorithm*	UTF-8 characters (no format checking)
1.3.6.1.4.1.1466.115.121.1.50	Telephone Number	Printable string (alphabetic, digits, **'**, **(**, **)**, **+**, **,**, **-**, **.**, **/**, **:**, **?**, and space) and **"**
1.3.6.1.4.1.1466.115.121.1.51	Teletex Terminal Identifier*	UTF-8 characters (no format checking)
1.3.6.1.4.1.1466.115.121.1.52	Telex Number*	Printable string (alphabetic, digits, **'**, **(**, **)**, **+**, **,**, **-**, **.**, **/**, **:**, **?**, space) and $ (no format checking)
1.3.6.1.4.1.1466.115.121.1.53	UTC Time	Like Generalized Time above, but with a two-digit year specification (*yy*)

Values defined using the binary and octet string syntaxes are transferred in binary and do not consist of UTF-8 characters. For syntaxes that have valid values of UTF-8 characters or IA5 characters, as shown in Table 34 on page 288, a value can include embedded hexadecimal 00's.

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 35. 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
1.3.6.1.4.1.1466.115.121.1.30	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
1.3.6.1.4.1.1466.115.121.1.54	LDAP Syntax Description
1.3.6.1.4.1.1466.115.121.1.58	Substring Assertion
1.3.18.0.2.8.1	IBM Attribute Type Description
1.3.18.0.2.8.3	IBM Entry UUID Description

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 are a fixed set as listed in the following table.

Table 36. Supported matching rules

Name	Numeric object identifier	Assertion syntax
Matching rules marked with a * are only supported when the LDAP server is running at server compatibility level 6 or higher.		
bitStringMatch*	2.5.13.16	Bit String. Must have same length and bits set.
booleanMatch	2.5.13.13	Boolean. Both values are either TRUE or FALSE. Case is ignored.
caseExactIA5Match	1.3.6.1.4.1.1466.109.114.1	IA5 String. Leading and trailing whitespace is ignored. Embedded whitespace is replaced by a single blank. Case must be the same.
caseExactMatch	2.5.13.5	Directory String. Leading and trailing whitespace is ignored. Embedded whitespace is replaced by a single blank. Case must be the same.
caseExactOrderingMatch	2.5.13.6	Directory String. Leading and trailing whitespace is ignored. Embedded whitespace is replaced by a single blank. Case must be the same. Collating sequence is based on the UTF-8 representation.
caseExactSubstringsMatch	2.5.13.7	Directory String. Leading and trailing whitespace is ignored. Embedded whitespace is replaced by a single blank. Case must be the same.
caseIgnoreIA5Match	1.3.6.1.4.1.1466.109.114.2	IA5 String. Leading and trailing whitespace is ignored. Embedded whitespace is replaced by a single blank. Case is ignored.
caseIgnoreIA5SubstringsMatch*	1.3.6.1.4.1.1466.109.114.3	IA5 String. Leading and trailing whitespace is ignored. Embedded whitespace is replaced by a single blank. Case is ignored.

Table 36. Supported matching rules (continued)

Name	Numeric object identifier	Assertion syntax
Matching rules marked with a * are only supported when the LDAP server is running at server compatibility level 6 or higher.		
caseIgnoreListMatch*	2.5.13.11	Postal Address. Leading and trailing whitespace is ignored. Embedded whitespace is replaced by a single blank. Case is ignored.
caseIgnoreListSubstringsMatch*	2.5.13.12	Postal Address. Leading and trailing whitespace is ignored. Embedded whitespace is replaced by a single blank. Case is ignored.
caseIgnoreMatch	2.5.13.2	Directory String. Leading and trailing whitespace is ignored. Embedded whitespace is replaced by a single blank. Case is ignored.
caseIgnoreOrderingMatch	2.5.13.3	Directory String. Leading and trailing whitespace is ignored. Embedded whitespace is replaced by a single blank. Case is ignored. Collating sequence is based on the UTF-8 representation.
caseIgnoreSubstringsMatch	2.5.13.4	Directory String. Leading and trailing whitespace is ignored. Embedded whitespace is replaced by a single blank. Case is ignored.
certificateExactMatch*	2.5.13.34	Certificate. Exact match of entire value.
certificateMatch*	2.5.13.35	Certificate. Exact match of entire value.
distinguishedNameMatch	2.5.13.1	Distinguished Name. Each name must have the same number of RDN components and each attribute within each RDN must match using the EQUALITY rule for that attribute type.
distinguishedNameOrderingMatch	1.3.18.0.2.4.405	Distinguished Name. The normalized string representation of each name is compared. The collating sequence is based on the UTF-8 representation.
generalizedTimeMatch	2.5.13.27	Generalized Time. The value is normalized as yyyymmddhhmmss.ffffffZ.
generalizedTimeOrderingMatch	2.5.13.28	Generalized Time. The value is normalized as yyyymmddhhmmss.ffffffZ.
IBM-EntryUUIDMatch	1.3.18.0.2.22.2	IBM Entry UUID. Hyphens are removed and a not case-sensitive string comparison is performed.
integerFirstComponentMatch	2.5.13.29	Integer.
integerMatch	2.5.13.14	Integer.
integerOrderingMatch*	2.5.13.15	Integer.
numericStringMatch*	2.5.13.8	Numeric String. Leading and trailing whitespace is ignored. Embedded whitespace is replaced by a single blank.
numericStringOrderingMatch*	2.5.13.9	Numeric String. Leading and trailing whitespace is ignored. Embedded whitespace is replaced by a single blank. Perform string order processing. Collating sequence is based on the UTF-8 representation.
numericStringSubstringsMatch*	2.5.13.10	Numeric String. Leading and trailing whitespace is ignored. Embedded whitespace is replaced by a single blank.

Table 36. Supported matching rules (continued)

Name	Numeric object identifier	Assertion syntax
Matching rules marked with a * are only supported when the LDAP server is running at server compatibility level 6 or higher.		
objectIdentifierMatch	2.5.13.0	Object Identifier. The value is normalized as an attribute descriptor.
objectIdentifierFirstComponentMatch	2.5.13.30	Object Identifier. The value is normalized as an attribute descriptor.
octetStringMatch	2.5.13.17	Octet String. Both values must contain the same number of octets and each octet must have the same value.
octetStringOrderingMatch*	2.5.13.18	Octet String. Perform string order processing. Collating sequence is based on the UTF-8 representation.
presentationAddressMatch*	2.5.13.22	Presentation Address. Leading and trailing whitespace is ignored. Embedded whitespace is replaced by a single blank. Case is ignored.
protocolInformationMatch*	2.5.13.24	Protocol Information. Leading and trailing whitespace is ignored. Embedded whitespace is replaced by a single blank. Case is ignored.
telephoneNumberMatch	2.5.13.20	Telephone Number
telephoneNumberSubstringsMatch	2.5.13.21	Telephone Number. The value is normalized using the **telephoneNumberMatch** rule.
uniqueMemberMatch*	2.5.13.23	Name And Optional UID. Each name must have the same number of RDN components and each attribute within each RDN must match using the EQUALITY rule for that attribute type. The optional UID is considered part of the rightmost RDN.
utcTimeMatch	2.5.13.25	UTC Time. The value is normalized as yyyymmddhhmmss.ffffffZ.

Notes on matching rules:

1. An undefined attribute type within a distinguished name uses the directory string matching rules.
2. The **aclEntry** and **entryOwner** attribute types use the distinguished name matching rules for **access-id, group**, and **role** scopes of protection. The assertion value is just the DN portion of the attribute value. The matching rules used by **aclEntry** and **entryOwner** attribute types for the **aclFilter** and **ownerFilter** scopes of protection are dependent on the filter attributes specified within the corresponding access control filter.
3. Attribute types with a binary transfer syntax cannot be used in a search filter but can be used in a compare operation.
4. The **ibm-allGroups** and **ibm-allMembers** attribute types cannot be used in a search filter. These are read-only operational attributes and results in a FALSE match status when used in a search filter.
5. The TDBM, LDBM, and CDBM backends ignore the **ORDERING** and **SUBSTR** matching rules and always uses the **EQUALITY** matching rule when processing a search filter.

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 might 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 is 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 must 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 might 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. If a plus sign ('**+**') is specified in the list of attributes to be returned, then all operational attributes other than **ibm-allMembers, ibm-allGroups, ibm-entryCheckSum, ibm-entryCheckSumOp**, and **hasSubordinates** are returned on the search response, if the user is authorized to read those operational attributes. 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*] [**RACFFIELD** *qdescrs*] **)**

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 437 for the definition of these values. The default attribute access class for an attribute is **normal**.

RACFFIELD *qdescrs*

The information needed to associate this attribute with a RACF custom field in a user or group profile.

The format of *qdescrs* is either a value in single quotation marks:

RACFFIELD '*racfFieldName***'**

or two values in parentheses, each in single quotation marks and separated by a blank:

RACFFIELD ('*racfFieldName***' '***racfFieldType***')**

racfFieldName format must be:

USER-CSDATA-*name*

or

GROUP-CSDATA-*name*

where *name* is the name of the associated RACF custom field.

racfFieldType is the type of custom field. The acceptable values are:
char, ***flag***, ***hex***, ***num***, and ***qchar***

This value defaults to **char** if the *racfFieldType* is not specified.

Note: When specifying **RACFFIELD**:

1. The syntax of the attribute type must be IA5 String (1.3.6.1.4.1.1466.115.121.1.26).
2. The same *racfFieldName* value cannot be specified in more than one **IBMAttributeTypes** schema definition.
3. The *racfFieldType* value can affect the processing of the attribute values.
4. See "Associating LDAP attributes to RACF custom fields" on page 338 for more information about the LDAP server support for RACF custom fields.

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 used by the LDAP server. Other IBM LDAP servers might also specify **DBNAME**, **LENGTH**, and other 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 are 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 might 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. 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. 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`

Search the site for “Private Enterprise Number” 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 OID arc `1.3.18.0.2.1000.100` is assigned. (**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
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
  )
ibmattributetypes: ( 1.3.18.0.2.1000.100.4.1 ACCESS-CLASS normal )
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
  )
ibmattributetypes: ( 1.3.18.0.2.1000.100.4.2 ACCESS-CLASS sensitive )
-
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 )
  )
-
```

This short description has described how to update the schema 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, the server supplies an initial schema. This initial schema is sufficient for usage of the SDBM (without RACF custom fields), CDBM (with configuration related entries), and GDBM backends, but must be updated for usage of LDBM, TDBM, SDBM with RACF custom fields, and CDBM with user-defined entries. The schema files shipped with the LDAP server, **schema.user.ldif** and **schema.IBM.ldif**, might be sufficient for your usage of LDBM, CDBM, or TDBM. See "Setting up the schema for LDBM, TDBM, and CDBM" on page 275 for more information about adding these files to the schema. If they are not sufficient, you can change the schema as needed. The schema entry is required and cannot be deleted. When in a sysplex, changes to the schema made at one LDAP server are broadcast to all the LDAP servers in the group. When deleting an attribute type or object class definition, you must provide just the object identifier enclosed in parentheses. Any additional fields that are specified are checked for appropriate syntax but are not used.

The operations supported include adding, modifying, or deleting any object class, attribute type, or IBM attribute type that is not part of the initial schema definition required by the LDAP server. Changes to the initial schema are restricted. See "Changing the initial schema" on page 299 for more information. The modifications (additions, changes, and deletions) specified by the LDAP modify

function are applied to the schema entry. The resulting schema entry becomes the active schema and is used by all backends to verify that directory changes adhere to it.

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 the schema might be specified in attribute type definitions.
- All attribute types referred to in IBM attribute type definitions must also be defined as attribute types.
- All structural object classes must include the **top** object class in their object class hierarchy.

Modifications to the schema are rejected if they would possibly make existing entries no longer valid. If there is an entry in a TDBM, CDBM, or LDBM backend that is using an attribute or object class:

- The attribute or object class cannot be deleted. Instead, "delete" the schema element by modifying it to mark it as **OBSOLETE** rather than deleting its definition from the schema entry. Therefore, no new entries can be created using the schema element and the existing entries which do use the schema element are still accessible. An existing entry that uses the **OBSOLETE** schema element must be modified to use only non-**OBSOLETE** schema elements during the next modification of the entry in order for the modification to succeed.
- The attribute or object class cannot be modified in a way that could affect the data in the entry. For example, the syntax of an attribute cannot be changed when that attribute is in use. You must modify the entries first so they do not use the object class or attribute, then change the schema.

  The following fields in an attribute type definition are the only fields that can be modified if the attribute type is in use by an entry:

  DESC
  OBSOLETE
  SINGLE-VALUE (can be removed but not added)
  NO-USER-MODIFICATION
  USAGE

  The following fields in an IBM attribute type definition can be modified:

  ACCESS-CLASS
  RACFFIELD

  The following fields in an object class definition can be modified when the object class is in use by an entry:

  DESC
  OBSOLETE
  MUST (can only move an attribute to **MAY**)
  MAY (can only add an attribute)

Changing the initial schema

The initial schema contains the **ldapSyntaxes, matchingRules, attributeTypes, IBMAttributeTypes**, and **objectClasses** needed by the LDAP server. See Appendix A, "Initial LDAP server schema," on page 633 for the contents of the initial schema.

The syntaxes, matching rules, attribute types, and IBM attribute types in the initial schema cannot be deleted or modified. The object classes in the initial schema cannot be deleted or modified, with the following exceptions:

1. **groupOfNames**
2. **groupOfUniqueNames**

These object classes allow the following fields to be modified:
DESC
MUST
MAY

The **MUST** and **MAY** lists can be modified in any way if no directory entries are using this object class. If there is a directory entry using this object class, the only **MUST** and **MAY** changes allowed are to move an attribute from the **MUST** list to the **MAY** list and to add an attribute to the **MAY** list.

Any part of a schema modification that attempts to add LDAP syntaxes or matching rules to the schema or to modify the initial schema except as described above is ignored, with no message issued to indicate this. The rest of the schema modification is performed and the result of those changes is returned to the client.

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: *Lightweight Directory Access Protocol (v3)* 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. Therefore, the schema file must 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. In order for the modification to succeed, the replacement values must include definitions for all schema definitions that are in use by existing directory entries and the replacement values must conform to the rules described above about what fields can be modified in an active schema entry.
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.

In all cases, the values of the attribute that are in the initial LDAP server schema cannot be deleted and can only be modified in limited ways as described in "Changing the initial schema" on page 299.

The behavior used by the LDAP server is selected in one of two ways:

1. Specify the **schemaReplaceByValue** option in the global section of the LDAP server configuration file to set the behavior for all modify with replace operations of the schema. Specifying **on** activates the schema-replace-by-value behavior; **off** activates the standard RFC behavior. See Chapter 8, "Customizing the LDAP server configuration," on page 83 for more information.
2. Specify the **IBMschemaReplaceByValueControl** 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. See Appendix C, "Supported server controls," on page 679 for more information.

If neither the **schemaReplaceByValue** configuration option or the **IBMschemaReplaceByValueControl** control is specified, the default behavior is schema-replace-by-value.

Example: assume that the objectclasses attribute for cn=schema 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' ... )
```

This is to replace `'oldObjectclass1'` and add a value for `'newObjectclass4'`.

This is the update file for the modify operation:

```
dn: cn=schema
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' ... )
```

IBM attribute types are extensions to the attribute type definition. The IBM attribute type is deleted when the corresponding attribute type is deleted. IBM attribute types are always replaced by value even when **schemaReplaceByValue off** is specified in the LDAP server configuration file. This ensures that access class protection is not inadvertently removed from an existing attribute type.

Updating a numeric object identifier (NOID)

It might become necessary to update the numeric object identifier (NOID) of an attribute type or object class 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 attribute type or object class whose NOID is to be changed; the value to add must specify the new NOID for the attribute type or object class, along with all the other parts of the attribute type or object class definition. For an attribute type, the **NAME, SUP, EQUALITY, ORDERING, SUBSTR**, and **SYNTAX** must be identical in the existing definition and the value to add. **SINGLE-VALUE** can be removed but not added. For an object class, **NAME, SUP, MUST, MAY**, and type (**ABSTRACT, STRUCTURAL**, or **AUXILIARY**) must be identical in the existing definition and the value to add. The entire attribute type or object class definition is replaced by the contents of the add. Note the object identifier assigned to an attribute type or object class cannot be changed if there are any directory entries using the attribute type or object class. Also, the object identifier of an attribute type or object class in the initial LDAP schema cannot be changed.

As an example for changing the NOID of the `xyz` attribute type from `1.3.5.7` to `2.4.6.8`, the update file for the modify operation is as follows:

```
cn=schema
-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 might be encountered when updating schema:

- For enhanced readability, *type:value* pairs in LDIF files might 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 must exist on the subsequent LDIF line. Various reason codes related to unrecognized values might be issued.
- Only limited changes are allowed to the initial schema, as described in "Changing the initial schema" on page 299. All other changes to the initial schema are ignored by the LDAP server with no error returned.
- The IBM attribute type schema attribute is an extension to the associated attribute type in the schema. If the schema update contains an IBM attribute type value for which an attribute type value is not defined, the schema update fails. 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 preferred. 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 Coordinated Universal Time syntax, use Coordinated Universal Time syntax in the search filter rather than local time. All time values are stored in the data store as Coordinated Universal Time 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. Note the scope must be **base** in the search request to display the schema.

```
ldapsearch -h ldaphost -p ldapport -s base -b "cn=schema" "objectclass=subschema"
```

Immediately after the server is started for the first time, this command produces the results shown in Appendix A, "Initial LDAP server schema," on page 633. After the schema is updated by an LDAP root or schema administrator, the search results show the full schema as the union of the initial schema and the added schema elements.

The search results contain these attributes:

```
cn=SCHEMA
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 LDAP server schema entry. To find the value of the **subschemaSubentry** attribute, specify **subschemaSubentry** as an attribute to be returned 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
```

Chapter 16. 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 topic 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 might result in unintentional or unwanted results.

In LDAP, modify DN operations are only supported in the TDBM (DB2-based), LDBM (file-based), and CDBM (file-based) backends.

Modify DN operation syntax

The z/OS implementation of the Modify DN operation supports all required and optional parameters that are described for the operation in RFC 2251: *Lightweight Directory Access Protocol (v3)*. Specifically, these parameters are required:

- **entryDN:** This is the Distinguished Name (DN) of the entry whose name is changed. This entry might not have subordinate entries. This parameter might not be a zero-length string.
- **newRdn:** The Relative Distinguished Name (RDN) that forms the leftmost component of the new name of the entry. This parameter might 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 are deleted from the entry.

The following parameter to the Modify DN operation is optional:

- **newSuperior:** The Distinguished Name (DN) of the entry that becomes the immediate superior of the renamed entry (identified by the **entryDN** parameter). If this parameter is present, it might consist of a zero-length string or a non-zero-length string. See “Modify DN operations related to suffix DNs” on page 315 for more information about the use of a zero-length string for this parameter. A zero-length string value for this parameter ("") signifies 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 C, “Supported server controls,” on page 679):

- **IBMModifyDNTimelimitControl:** This control causes the Modify DN operation to be abandoned if its duration exceeds the time limit that is represented by the control value that is expressed in seconds. No changes are made if the operation is abandoned. This control is accepted even if it is set by the admin DN for the server. When this control is present, it does not propagate to the replica servers. (See “Modify DN operations and replication” on page 322 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 that construct DN-type attribute values (those whose attribute syntax is not distinguished name but, which might be used to store DN values). They are **aclEntry** and ownership **entryOwner** attributes. Updates to constructed DN types are limited to these two attributes that are defined by the LDAP Server. No changes are 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 LDBM, TDBM, or CDBM backend. Users cannot limit the scope of values that 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. Note that even if the control is designated as non-critical, the server still tries to accept the intent of the control and if this attempt fails, the entire Modify DN operation fails.

  When **IBMModifyDNRealignDNAttributesControl** is present on a request to a master server on which replication of Modify DN operations is enabled, it is propagated to the replica servers. (See "Modify DN operations and replication" on page 322 for more information about replication of Modify DN operations.)

A few simple examples of the use of the Modify DN operation follow. Each request is expressed in the format of the ModifyDNRequest defined in RFC 2251: *Lightweight Directory Access Protocol (v3)* and in the corresponding invocation command for the z/OS client utility program **ldapmodrdn**. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information about 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"
```

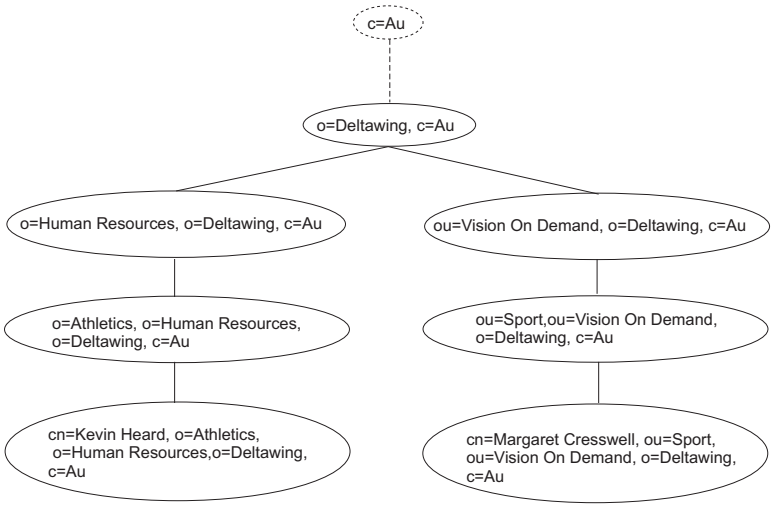


Figure 10. Before Modify DN operation

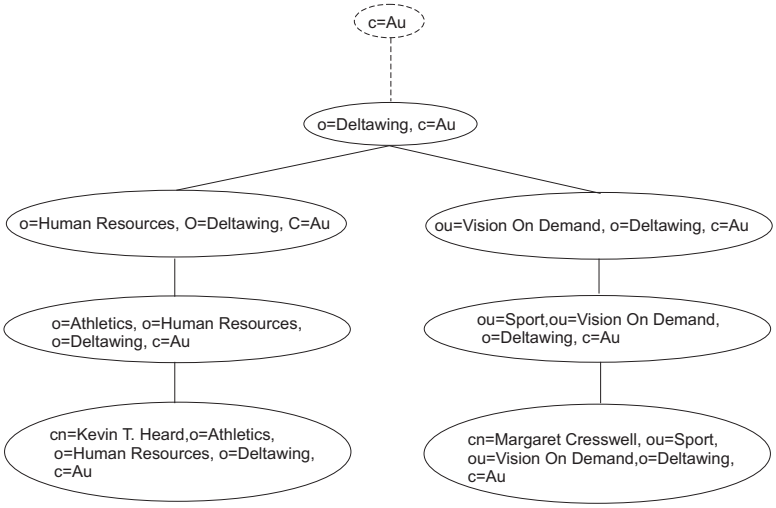


Figure 11. After Modify DN operation

Note: The **-r** parameter specifies that the old RDN attribute value (`cn=Kevin Heard`) is 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."
```

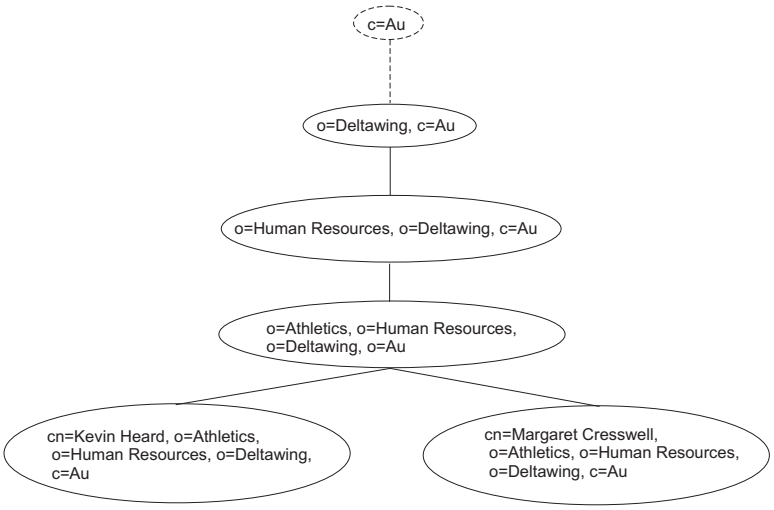


Figure 12. Before Modify DN operation

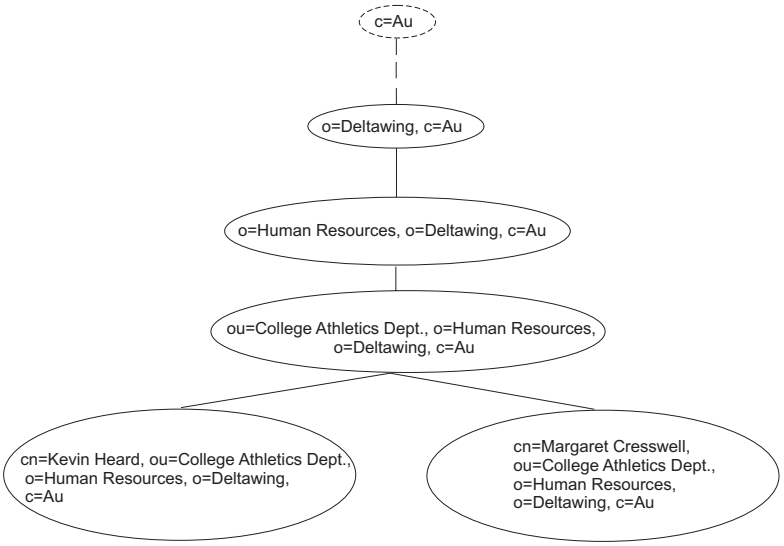


Figure 13. After Modify DN operation

Note: The absence of the **-r** parameter specifies that the old RDN attribute value (`o=Athletics`) is 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"
```

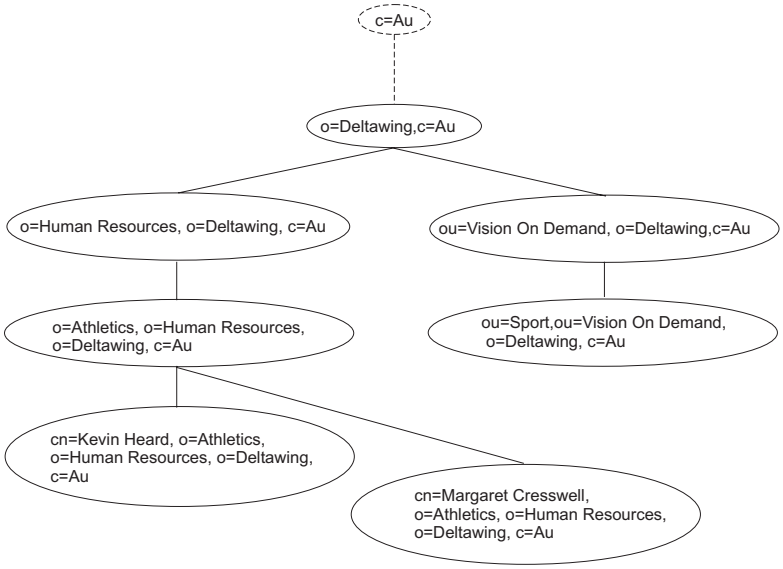


Figure 14. Before Modify DN operation

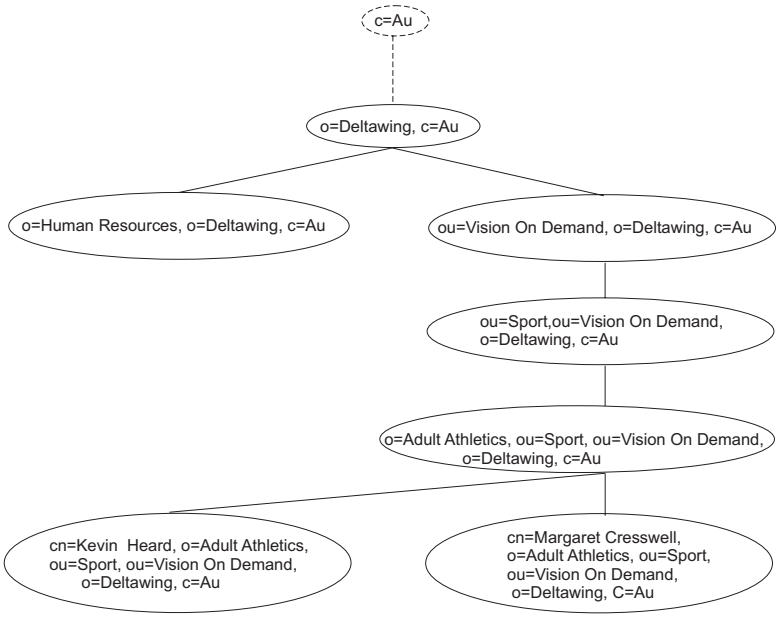


Figure 15. After Modify DN operation

Note: The absence of the **-r** parameter specifies that the old RDN attribute value (`o=Athletics`) is preserved in the target entry after this operation. The target entry and descendants in its subtree are 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 comprehend the data before using the Modify DN operation. Specifically, the user must 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 might result in modification of additional directory entries which are outside the scope of the directory subtrees being moved. This topic explains and gives 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 might result in a long-running transaction, which might 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 309 for more information.
- The scope of the changes might result in unanticipated effects in the directory and might affect user access to these entries. See "Access control changes" on page 312 for more information.
- There are limitations to which directory entries are eligible for the Modify DN operation. See "Eligibility of entries for rename" for more information.
- In case the directory needs to be returned to a state before a Modify DN operation, the directory should be backed up by using the **ds2ldif** utility program or, for a TDBM backend, by using DB2 utilities to generate a DB2 image copy of the underlying table spaces. See Chapter 12, "Running and using the LDAP server utilities," on page 217 for more information about the **ds2ldif** utility program, and IBM Information Management Software for z/OS Solutions Information Center for more information about the DB2 image copy.
- There are considerations if the data to be modified by this operation is being replicated. See "Modify DN operations and replication" on page 322 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 in the same TDBM, LDBM, or CDBM backend targeted by the Modify DN operation. The Modify DN operation with *newSuperior* option moves subtree entries within the same TDBM, LDBM, or CDBM backend, and does not permit movement of subtree entries from one backend to another. The entry to be renamed must exist in the backend, and the new DN for the entry must not exist in the backend.
2. Referral entries might 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 the referral server 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 that refer to any of the entries whose DNs were modified in the local directory by a Modify DN operation must be manually updated to reflect the changes; no automatic updates are propagated to those servers from the local one.
3. The LDAP server schema entry cannot be renamed.
4. Entries renamed by a Modify DN operation must conform to the LDAP server schema. As such, the RDN attribute type must be consistent with the schema rules for the object classes of the entry: a Modify DN operation fails if the attribute type of *newRdn* is not in the **MUST** or **MAY** list for the entry's object classes.
5. If a new superior entry is specified, it must be in the same backend as the entry to be renamed but may be under a different suffix managed by that backend. If the **IBMModifyDNRealignDNAttributesControl** is specified, only entries within the same backend as the renamed entry are processed.

6. When **IBMModifyDNRealignDNAttributesControl** is present on a Modify DN request, the operation looks for occurrences of each renamed DN (this can be multiple DNs if renaming a subtree) in certain attributes within all the entries in the backend and replaces each renamed DN with its new DN. The affected attributes are:
   a. Any attribute whose syntax is DN syntax (OID 1.3.6.1.4.1.1466.115.121.1.12).
   b. The **aclEntry** and **entryOwner** attributes (these contain DNs in a structured format).
7. If *newRdn* is specified on a Modify DN operation, each attribute in the *newRdn* value is added to the entry when it is moved. If a *newRdn* attribute already has a different value in the entry and the attribute is defined as **SINGLE-VALUE** in the schema, the Modify DN operation fails. 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 **SINGLE-VALUE** attribute **sector** with a value of `northwest`. This rename fails because it attempts to add a second value (`northeast`) to the **sector** attribute.

   If the *newRdn* attribute is contained in the current RDN, then the *deleteoldrdn* parameter can be added to the Modify DN operation to allow it to succeed. In this case, the current attribute value is removed so that the attribute only contains the one value from *newRdn* in the renamed entry. For example, suppose an entry with DN of `sector=northwest,ou=mydivision,o=MyCompany,c=us` is to be renamed with the *newRdn* `sector=northeast` and *deleteoldrdn* is specified on the Modify DN operation. This rename succeeds because `northwest` is replaced by `northeast` as the single value of the **sector** attribute in the renamed entry.
8. Entries might 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 310 for detailed explanation and examples of this effect.
9. 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 if this does not result in an **aliasedObjectName** value that is a DN equal to the DN of the renamed alias entry.
10. When advanced replication is configured, a Modify DN operation from one replication context to a different replication context is not supported. The Modify DN operation must occur within the same replication context.
11. The global password policy entry, **cn=pwdpolicy,cn=ibmpolicies**, cannot be renamed. If a user or group entry has a reference to a password policy entry in an **ibm-pwdIndividualPolicyDN** or **ibm-pwdGroupPolicyDN** attribute value, the individual or group password policy entry cannot be renamed or deleted until the association is removed from all user or group entries.

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 might 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 might 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 can process concurrent LDAP operations targeted at a given TDBM, LDBM, or CDBM backend while the Modify DN operation is in progress, the extent to which such concurrency is possible depends on what data in the directory might be needed and locked by the competing operations. In addition, if the number of entries being affected by the TDBM Modify DN operation is large or if the database is small, the underlying DB2 locking mechanism might 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 might 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 at “Modify DN operation syntax” on page 303 would 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 about DB2 lock escalation, see IBM Redbook, *Locking in DB2 for MVS/ESA Environment (SG24–4725)*, at http://www.redbooks.ibm.com or IBM Information Management Software for z/OS Solutions Information Center.

Access control and ownership

For all entries being renamed, the caller must have **w**(rite) permissions for the attribute values that must 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. Note 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 that our sample directory contains the following entry, which is the target of a Modify DN operation, and that 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 is the new Superior 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 and 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

If 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 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 that is 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 inherits any propagating ACLs applying to the *newSuperior* entry. As a consequence, entries to which a user had access before the request might no longer be accessible by that user, and entries to which access was denied for a given user before the request is 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 are modified. Therefore, before such a move or rename, users should

carefully consider how ownership and accessibility to entries protected by these attributes might change after the move, and what ACL and ownership changes might be wanted, 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"
```

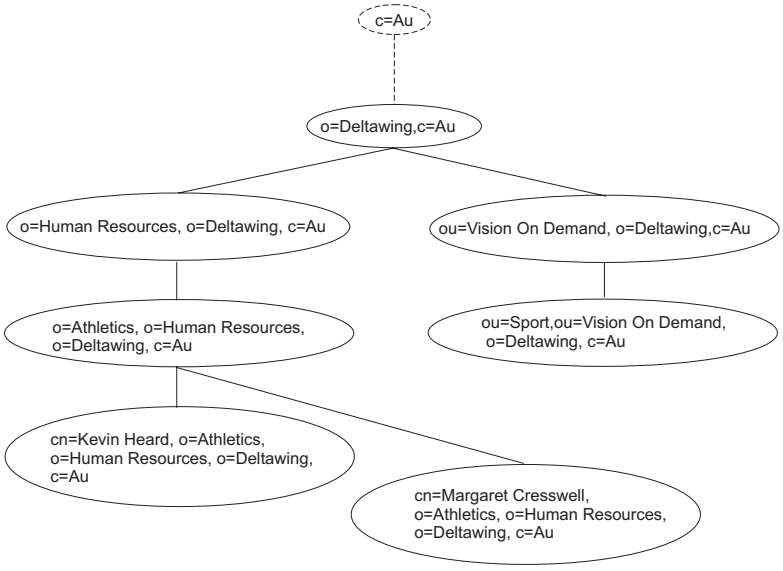


Figure 16. Before Modify Dn operation

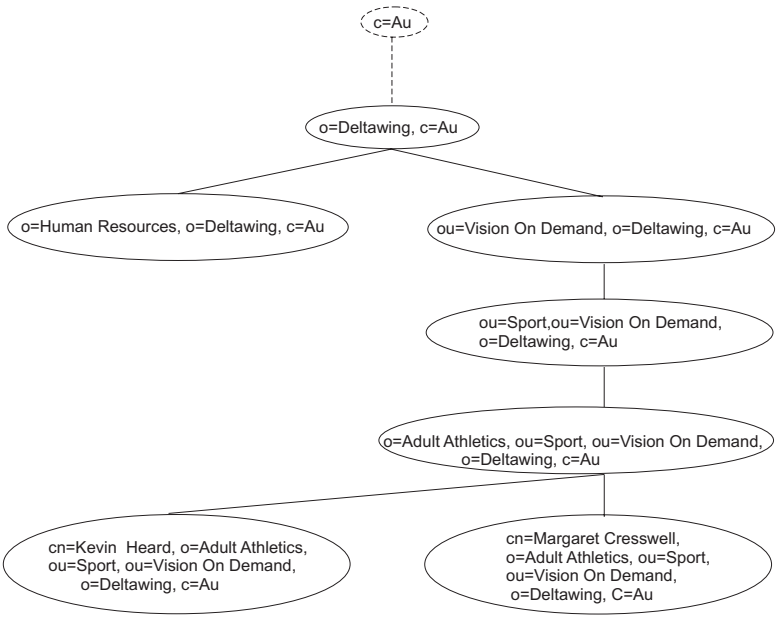


Figure 17. After Modify DN operation

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 that 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 has only **r** access on both sensitive and critical attributes in the entry after the relocation. It might be expected that a given DN has 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 want to return the moved entry (and its subordinates) back to their original location in the directory hierarchy, this is not 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 affect 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. Therefore, while it might 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 before the relocation preserves that explicit ownership after the relocation is performed. Any entries in the relocated subtree which inherited ownership before relocation continues to inherit ownership following relocation. If the owning entry before relocation was a node superior to the relocated entry, the owning entry is 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 before relocation continue to propagate ownership to subordinates after the relocation.

See the example in "Access control changes" on page 312.

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`. Therefore, the act of relocating an entry might change the effective owner of that entry and of its subordinates.

Modify DN operations related to suffix DNs

The Modify DN operation might be used to modify the DNs of any entries in a TDBM or LDBM 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 might also be renamed. Suffix entries might be renamed to become non-suffix entries and suffix entries might be renamed such that they continue to be suffix entries. In addition, non-suffix entries might be renamed to become suffix entries. This section provides example scenarios for rename operations which involve suffix entries. It summarizes constraints which are adopted for the LDAP directory implementation which are not defined in the protocol behavior prescribed by RFC 2251: *Lightweight Directory Access Protocol (v3)* for the Modify DN operation. Examples are provided on how various renaming scenarios might be accomplished, and factors to be considered when performing these operations are mentioned.

Note: Do not rename the **cn=ibmpolicies** and **cn=configuration** CDBM suffix entries. Renaming these suffixes can cause configuration related problems. The global password policy entry, **cn=pwdpolicy,cn=ibmpolicies**, cannot be renamed. If a user or group entry has a reference to a password policy entry in an **ibm-pwdIndividualPolicyDN** or **ibm-pwdGroupPolicyDN** attribute value, the individual or group password policy entry cannot be renamed or deleted until the association is removed from all user or group entries.

Scenario constraints

Several constraints apply which are not defined by RFC 2251: *Lightweight Directory Access Protocol (v3)* in the description of the protocol behavior:

1. If an entry being renamed becomes (or remains) a suffix, the new DN must be designated in the server's configuration file as a suffix for the backend, otherwise the operation is not permitted.
2. The *newRdn* parameter of the Modify DN request must contain a non-null value, otherwise the operation request is treated as an error.
3. If the *newSuperior* parameter is present, it might contain a zero-length string signifying that the new entry does not have a superior entry, therefore is a suffix entry.

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.

   For 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 entry
   `ou=End_GPL, o=MyCompany, c=US`
   to suffix entry
   `ou=Endicott, o=MyCompany, c=US`

   The following figure shows an example of this operation:

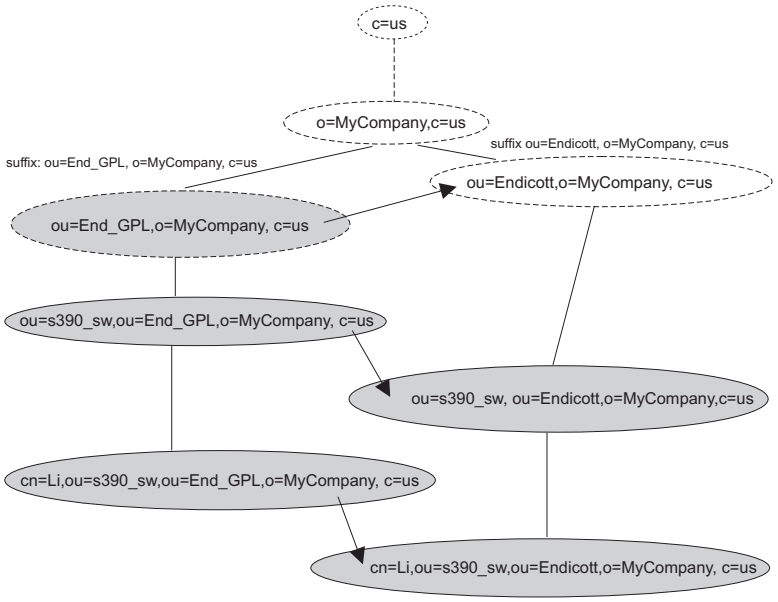


Figure 18. Suffix rename with no new superior

The new DN must be already designated as a suffix for this backend, otherwise this operation fails.

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. For 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 entry
   o=MyCompany, c=us

   The following figure shows an example of this operation:

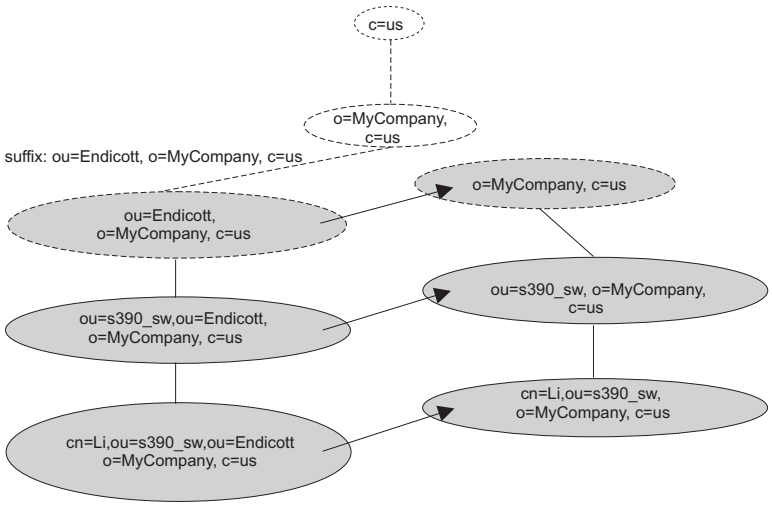


Figure 19. Suffix rename with new superior

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. 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 target suffix that you want 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 requested suffix o=MyCompany, c=us, 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. For 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 entry:

   div=S390, ou=Endicott, o=MyCompany, c=us

   The following figure shows an example of this operation:

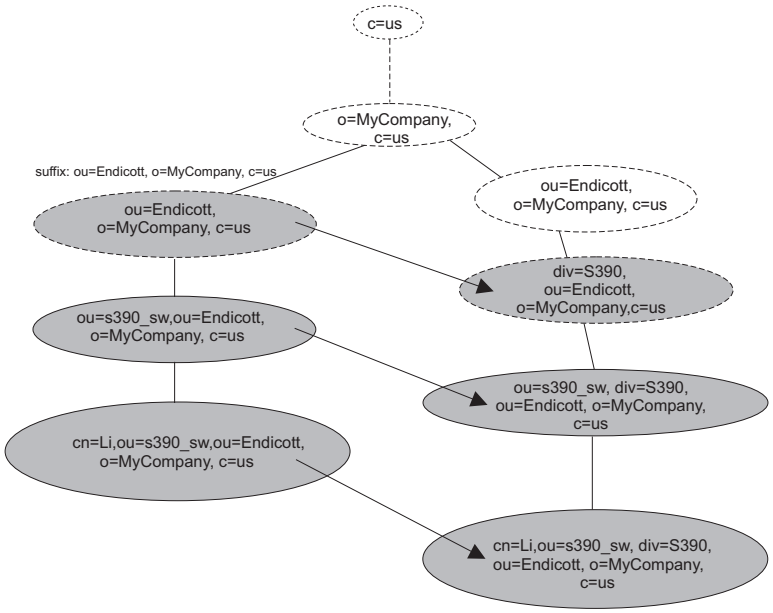


Figure 20. Overlapping suffix rename A

This rename can be accomplished by having a temporary suffix pre-defined for this backend in the 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 as follows:

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`

If basic replication is configured, 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 this must be followed to permit this. Advanced replication does not support a Modify DN operation from one replication context to another replication context. Therefore, the following procedure only works with basic replication.

a. Stop the replica server, add the temporary suffix (`o=OurTemporarySuffix` in our examples), restart the replica server.
b. On the master server, perform the previous Steps 3a and 3b from the examples above. This results in the intermediate rename to be performed on the master server and the results to be propagated to the replica server.
c. Stop the replica server, 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.

d. On the master server, perform the previous Step 3c on page 319 from the examples above. This results in the rename of entries to the final destination on the master server and in the results being propagated to the replica server.

4. Rename of suffix DN (some component other than RDN), and the new DN remains a suffix after the rename is completed. For 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:
   ```
   ou=Endicott, o=MyCompany, c=us
   ```
   to suffix entry `ou=Endicott, o=MyCompany_ny, c=us`

   The following figure shows an example of this operation:

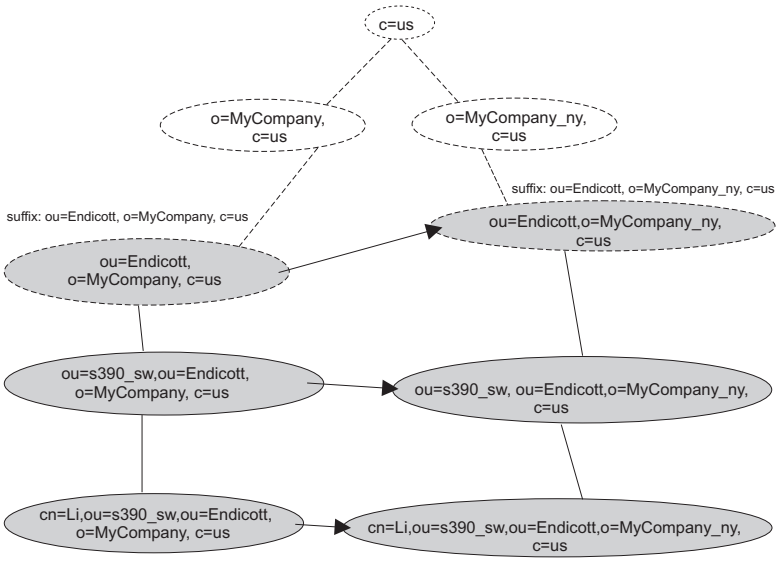


Figure 21. Overlapping suffix rename B

   The new DN must be already designated as a suffix for this backend, otherwise this operation fails. The operation is performed the same as a rename of any other DN in the directory. The product permits the rename to occur in one step, even if an entry for *newSuperior* does not already exist, since the newly named entry becomes 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. For 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 ou=End, o=MyCompany, c=us to non-suffix entry ou=GPL, ou=End, ou=MyCompany_na, o=MyCompany, c=us

The following figure shows an example of this operation:

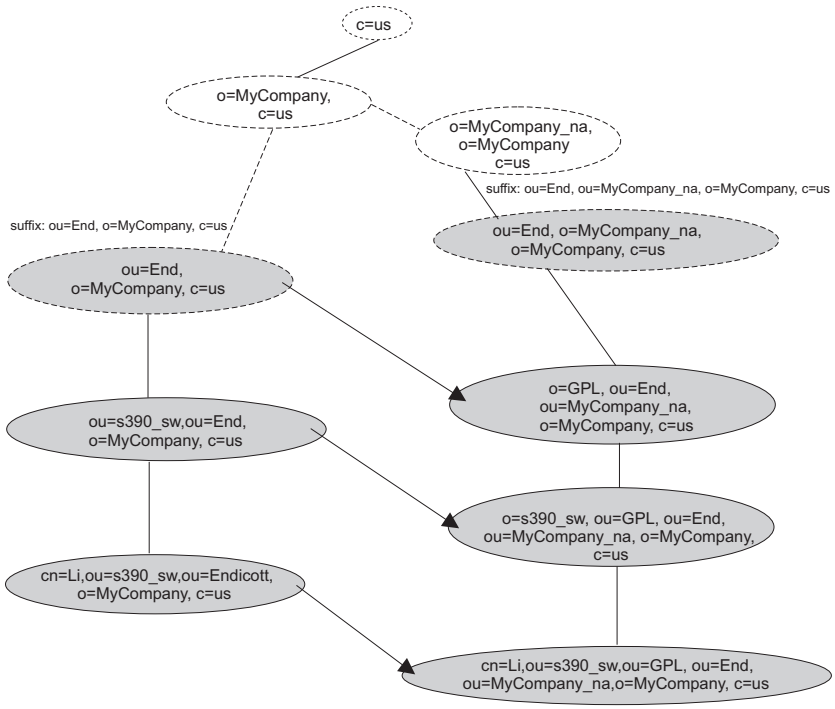


Figure 22. Suffix rename to non-suffix entry

The *newSuperior* entry must already exist before this operation is permitted.

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. For 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 an example of this operation:

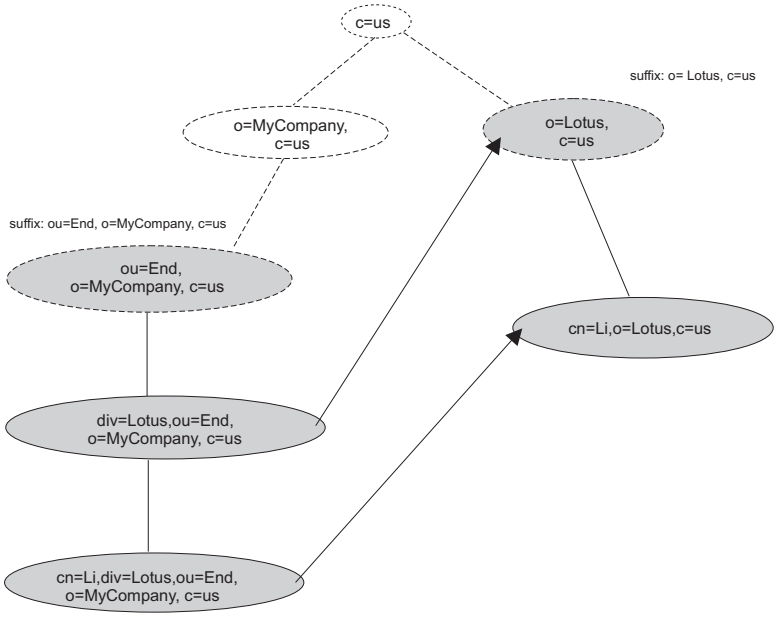


Figure 23. Rename non-suffix entry to suffix entry

The new DN must be already designated as a suffix for this backend, otherwise this operation fails.

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 that rename a leaf node, and that are not accompanied by the *newSuperior* parameter or the **IBMModifyDNRealignDNAttributesControl** control or the **IBMModifyDNTimelimitControl** control.
2. Complex Modify DN operations are those that either rename a mid-tree (non-leaf) node, or that are accompanied by the *newSuperior* parameter, or that are accompanied by either the **IBMModifyDNRealignDNAttributesControl** control or the **IBMModifyDNTimelimitControl** control.

If basic replication is configured, simple Modify DN operations are always accepted by the master server, and are replicated if replica entries are present in the TDBM, LDBM, or CDBM backend where a Modify DN operation is applied.

If advanced replication is configured, simple Modify DN operations are only accepted by the supplier server when the operation occurs within the same replication context.

A compatible server version is one known to support for Modify DN operations all features and controls implemented by the z/OS LDAP server including:

- the **IBMModifyDNRealignDNAttributesControl** control

- the **IBMModifyDNTimelimitControl** 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 or consumer server, all complex Modify DN operations are refused at the master server. If advanced replication is configured, complex Modify DN operations are only allowed within the same replication context.

Initial validation of compatible server versions in consumer and replica servers

Checks are made of consumer or replica servers by the supplier or master server which are intended to increase the likelihood that complex Modify DN operations will be successfully replicated.

The LDAP server must be able to establish a connection to each of the consumer or replica servers represented by replication agreement entries or replica entries in a TDBM or LDBM backend. When the connection is established to a given consumer or replica server, the supplier or master server determines if the consumer or 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 consumer or 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 on the consumer or master server. If a connection is established to a consumer or replica server and it is determined that the consumer or replica is not at a compatible server version, complex Modify DN operations are refused at the consumer or master server. In a basic replication environment, the replication of simple Modify DN operations is always permitted, and such operations are always performed at the master server. In an advanced replication environment, simple and complex Modify DN operations must occur within the same replication context, otherwise they are not allowed.

Periodic validation of compatible server versions in basic replication replicas

The following periodic replica checks are only performed in a basic replication environment and not in an advanced replication environment.

The master server may enable or disable processing of complex Modify DN operations, depending on dynamically changing states of replica servers and of replica entries within the master server's TDBM or LDBM 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 replica entries in the TDBM or LDBM backend, and if a connection can no longer be established to any 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 replica entries
- deletion of existing replica entries
- modification of existing replica entries in the TDBM or LDBM 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.

To determine whether a replica server is at a compatible version level, submit a root DSE search to that server, similar to the following. The **-D** and **-w** options only need to be specified if the replica server does not support anonymous binds.

```
ldapsearch -h ldaphost -p ldapport -D binddn -w passwd
 -s base -b "" objectclass=* ibm-enabledCapabilities
```

where *ldaphost* represents the host name on which the replica server runs, *ldapport* is the port number on which the replica server is listening, and *binddn* and *passwd* are the distinguished name and password of a user on the replica server.

If the **ibm-enabledCapabilities** attribute is returned on the root DSE search and its values contain `1.3.18.0.2.32.33` (subtree move) or `1.3.18.0.2.32.34` (subtree rename), then the replica server can support those operations.

Loss of basic replication synchronization because of incompatible replica server versions

The LDAP server basic 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 basic replication process could stall and the synchronization between the master server and its replicas could be lost.

Attention

It is suggested that an LDAP administrator responsible for configuring replication ensure that each replica server is at a compatible server version level before starting a master server which may be the recipient of complex Modify DN operations.

Loss of basic replication synchronization because of incompatible replica server versions - recovery

If at some point a master server accepts a complex Modify DN operation which cannot 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 replica entry 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 you want to later restart the offending replica (such as, after it has been started 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 that the two directories are synchronized.

Chapter 17. Accessing RACF information

RACF provides definitions of users, groups, classes, and general resources, and access control for resources. The LDAP server can provide LDAP access to this information stored in RACF.

Using SDBM, the RACF database backend of the LDAP server, you can:
- Add, modify, and delete RACF users, groups, and general resources. Note that data set resources are not supported.
- Add, modify, and delete user connections to groups
- Add and remove users and groups in general resource access lists
- Modify SETROPTS options that affect classes (for example, RACLIST)
- Retrieve RACF information for users, groups, connections, general resources, and class options
- Retrieve RACF user password and password phrase envelopes

The SDBM backend of the LDAP server implements portions of the **adduser, addgroup, rdefine, altuser, altgroup, ralter, permit, setropts, deluser, delgroup, rdelete, connect, remove**, and **search** RACF commands. SDBM uses the **R_admin** "run command" interface to start these RACF commands. As a result, this support is subject to the restrictions and authorization requirements of the **R_admin** interface. See *z/OS Security Server RACF Callable Services* for more information about these topics. One restriction in particular affects return of search results obtained by using the RACF **search** command. See "RACF restriction on amount of output" on page 356 for more details.

SDBM uses the **R_admin** profile extract functions to retrieve user, group, connection, and resource information. It uses the **R_admin** setropts extract function to retrieve class options information. These interfaces are not subject to any restrictions on the amount of data returned, but they do require appropriate authorization.

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 used 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 60 for information about getting your LDAP server configured with SDBM.

SDBM authorization

SDBM operations can be performed after several different types of binds to the LDAP server. In each of these binds, the LDAP server associates a RACF user ID with the bound user. SDBM starts RACF commands under the context of this RACF user ID, and RACF uses its normal authorization processing to determine what this RACF user ID can do.

The supported bind mechanisms are:

- Simple bind to SDBM: The RACF user ID is specified in the bind DN. See "Binding using a RACF user ID and password or password phrase" for more information.
- LDBM, TDBM, or CDBM native authentication bind: The RACF user ID specified in the native authentication entry is used. See Chapter 20, "Native authentication," on page 403 for more information.
- Kerberos bind: The RACF user ID is mapped by SDBM from the Kerberos identity. See "SDBM mapping" on page 397 for more information.
- Certificate bind: The RACF user ID associated with the certificate is used. See "Support of certificate bind" on page 76 for more information.

Binding using a RACF user ID and password or password phrase

The SDBM backend allows for directory authentication (or bind) using the RACF user ID and password or password phrase. 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 by using RACF access support but having a non-RACF security manager succeeds if the RACROUTE REQUEST=VERIFY,ENVIR=CREATE macro function call made by the LDAP server is successful. However, group membership information might not be available for the bound distinguished name if the security manager is not RACF.
- An LDAP simple bind that is made to a z/OS LDAP server by using RACF access support provides a successful or unsuccessful LDAP return code. In addition, if the LDAP return code 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 LDAP reason codes are mapped to return codes returned by the RACROUTE REQUEST=VERIFY,ENVIR=CREATE macro:

Table 37. LDAP return and reason codes returned to the client when binding to SDBM

LDAP return code	Reason code	Text
LDAP_INVALID_CREDENTIALS	R000104	The password is not correct or the user ID is not completely defined (missing password or uid)
LDAP_INVALID_CREDENTIALS	R000105	A bind argument is not valid
LDAP_INVALID_CREDENTIALS	R000100	The password expired
LDAP_INVALID_CREDENTIALS	R000101	The new password is not valid
LDAP_INVALID_CREDENTIALS	R000102	The user ID has been revoked
LDAP_OPERATIONS_ERROR	R000208	Unexpected racroute error safRC=*safRC* racfRC=*racfRC* racfReason=*racfReason*

Note:

1. The same reason codes are issued when binding using a password or a password phrase.

2. The use of RACF passtickets is supported by the z/OS LDAP server when binding through SDBM. The job name associated with the LDAP Server started task should be used as the application name when generating RACF passtickets. See *z/OS Security Server RACF Macros and Interfaces* for more information about RACF passtickets.

Binding with SDBM using password policy

When authenticating with a user in the SDBM backend, the password policy applied is determined by the underlying z/OS Security Server. Therefore, any configured z/OS LDAP password policy does not apply in these scenarios.

When the **PasswordPolicy** server control is sent on a bind request, the **PasswordPolicy** response control is returned on the bind response and has additional warning and error information about the authenticating user's password. Based on information returned from the Security Server, the SDBM backend only supports the following **PasswordPolicy** response control error codes during bind: **accountLocked, insufficientPasswordQuality, mustSupplyOldPassword**, and **passwordExpired**. See "PasswordPolicy" on page 684 for more information.

SDBM group gathering

After successfully authenticating to the LDAP server, a list is created of the groups to which the authenticated RACF user ID belongs. Only groups in which the user ID's membership is active (has not been revoked) are included in the list. This group membership list is used in authorization checking when trying to access entries in directories on the LDAP server.

If the SDBM backend is to be used for authentication purposes only and group membership is not needed, consider having your clients use the **authenticateOnly** server control, to streamline bind processing. This control overrides any extended group membership searching and default group membership gathering and is supported for Version 3 clients. See Appendix C, "Supported server controls," on page 679 for more information.

Note that the **authenticateOnly** control is not necessary if there is no TDBM, LDBM, GDBM, or CDBM backend configured. In this case, SDBM does not do any group gathering.

Associating LDAP attributes to RACF fields

Each RACF field in a user, group, connection, and resource profile and in the RACF class options must be associated with an LDAP attribute. The LDAP attribute is used to set the RACF field value in LDAP add and modify operations and to represent the RACF field in LDAP search output. There are two types of RACF fields:

- The fixed fields are defined by RACF. For each profile, these fields make up all the segments supported by SDBM (including the base segment) except the CSDATA segment.
- The custom fields are defined by customers. These fields make up the CSDATA segment in the profile.

Each of these types is associated to an LDAP attribute in a different way.

Associating LDAP attributes to RACF fixed fields

The fields defined by RACF for user, group, connection, and resource profiles, and for class options (setropts) are mapped to predefined attributes in the LDAP schema. These LDAP attributes cannot be deleted or modified and the attribute names cannot be changed. The following tables show the RACF fixed field names and the associated LDAP attribute names for user profiles (Table 38), group profiles (Table 39 on page 333), connection profiles (Table 40 on page 334), resource profiles (Table 41 on page 334) and setropts (Table 42 on page 337). The RACF names in the table are the keywords used to set the field in RACF commands or used by RACF in display output (for display-only fields). Not all names apply to all versions of LDAP and RACF.

Table 38. Mapping of LDAP attribute names to RACF fixed fields (user)

RACF segment name	RACF keyword in altuser/adduser/listuser	LDAP attribute name
User base	ADDCATEGORY	racfSecurityCategoryList
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	AUTH not displayed by LDAP	racfConnectGroupAuthority
User base	CLAUTH	racfClassName
User base	DFLTGRP	racfDefaultGroup
User base	GROUP	racfConnectGroupName
User base	Not modifiable - displayed as LAST-ACCESS	racfLastAccess
User base	NAME	racfProgrammerName
User base	Not modifiable - displayed as PASSDATE	racfPasswordChangeDate
User base	Not modifiable - displayed as PASS-INTERVAL	racfPasswordInterval
User base	PASSWORD	racfPassword
User base	password envelope - not modifiable	racfPasswordEnvelope
User base	Not modifiable - displayed as PASSWORD ENVELOPED	racfHavePasswordEnvelope
User base	password phrase envelope - not modifiable	racfPassPhraseEnvelope
User base	PHRASE	racfPassPhrase
User base	Not modifiable - displayed as PHRASEDATE	racfPassPhraseChangeDate
User base	Not modifiable - displayed as PHRASE ENVELOPED	racfHavePassPhraseEnvelope
User base	RESUME	racfResumeDate
User base	REVOKE	racfRevokeDate
User base	SECLABEL	racfSecurityLabel
User base	SECLEVEL	racfSecurityLevel

Table 38. Mapping of LDAP attribute names to RACF fixed fields (user) (continued)

RACF segment name	RACF keyword in altuser/adduser/listuser	LDAP attribute name
User base	UACC - value is not displayed by LDAP	racfConnectGroupUACC
User base	WHEN(DAYS())	racfLogonDays
User base	WHEN(TIME())	racfLogonTime
User base or Group base	Not modifiable - displayed as CREATED	racfAuthorizationDate
User base or Group base	DATA	racfInstallationData
User base or Group base	MODEL	racfDatasetModel
User base or Group base	OWNER	racfOwner
CICS® segment	OPCLASS	racfOperatorClass
CICS segment	OPIDENT	racfOperatorIdentification
CICS segment	OPPRTY	racfOperatorPriority
CICS segment	RSLKEY	racfRslKey
CICS segment	TIMEOUT	racfTerminalTimeout
CICS segment	TSLKEY	racfTslKey
CICS segment	XRFSOFF	racfOperatorReSignon
DCE segment	AUTOLOGIN	racfDCEAutoLogin
DCE segment	DCENAME	racfDCEPrincipal
DCE segment	HOMECELL	racfDCEHomeCell
DCE segment	HOMEUUID	racfDCEHomeCellUUID
DCE segment	UUID	racfDCEUUID
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
EIM segment	LDAPPROF	racfLDAPProf
KERB segment	ENCRYPT	racfEncryptType
KERB segment	KERBNAME	krbPrincipalName
KERB segment	Not modifiable - displayed as KEY FROM	racfKerbKeyFrom
KERB segment	Not modifiable - displayed as KEY VERSION	racfCurKeyVersion
KERB segment	MAXTKTLFE	maxTicketAge
LANGUAGE segment	PRIMARY	racfPrimaryLanguage
LANGUAGE segment	SECONDARY	racfSecondaryLanguage
LNOTES segment	SNAME	racfLNotesShortName
NDS segment	UNAME	racfNDSUserName
NETVIEW segment	CONSNAME	racfDefaultConsoleName

Table 38. Mapping of LDAP attribute names to RACF fixed fields (user) (continued)

RACF segment name	RACF keyword in altuser/adduser/listuser	LDAP attribute name
NETVIEW segment	CTL	racfCTLKeyword
NETVIEW segment	DOMAINS	racfDomains
NETVIEW segment	IC	racfNetviewInitialCommand
NETVIEW segment	MSGRECVR	racfMSGRCVRKeyword
NETVIEW segment	NGMFADMN	racfNGMFADMKeyword
NETVIEW segment	NGMFVSPN	racfNGMFVSPNKeyword
NETVIEW segment	OPCLASS	racfNetviewOperatorClass
User OMVS segment	ASSIZEMAX	racfOmvsMaximumAddressSpaceSize
User OMVS segment	CPUTIMEMAX	racfOmvsMaximumCPUTime
User OMVS segment	FILEPROCMAX	racfOmvsMaximumFilesPerProcess
User OMVS segment	HOME	racfOmvsHome
User OMVS segment	MEMLIMIT	racfOmvsMemoryLimit
User OMVS segment	MMAPAREAMAX	racfOmvsMaximumMemoryMapArea
User OMVS segment	PROCUSERMAX	racfOmvsMaximumProcessesPerUID
User OMVS segment	PROGRAM	racfOmvsInitialProgram
User OMVS segment	SHARED, AUTOUID	racfOmvsUidKeyword
User OMVS segment	SHMEMMAX	racfOmvsSharedMemoryMaximum
User OMVS segment	THREADSMAX	racfOmvsMaximumThreadsPerProcess
User OMVS segment	UID	racfOmvsUid
OPERPARM segment	ALTGRP	racfAltGroupKeyword
OPERPARM segment	AUTH	racfAuthKeyword
OPERPARM segment	AUTO	racfAutoKeyword
OPERPARM segment	CMDSYS	racfCMDSYSKeyword
OPERPARM segment	DOM	racfDOMKeyword
OPERPARM segment	HC	racfHcKeyword
OPERPARM segment	INTIDS	racfIntidsKeyword
OPERPARM segment	KEY	racfKEYKeyword
OPERPARM segment	LEVEL	racfLevelKeyword
OPERPARM segment	LOGCMDRESP	racfLogCommandResponseKeyword
OPERPARM segment	MFORM	racfMformKeyword
OPERPARM segment	MIGID	racfMGIDKeyword
OPERPARM segment	MONITOR	racfMonitorKeyword
OPERPARM segment	MSCOPE	racfMscopeSystems
OPERPARM segment	ROUTCODE	racfRoutcodeKeyword
OPERPARM segment	STORAGE	racfStorageKeyword
OPERPARM segment	UD	racfUDKeyword
OPERPARM segment	UNKNIDS	racfUnknidsKeyword
User OVM segment	FSROOT	racfOvmFileSystemRoot
User OVM segment	HOME	racfOvmHome

Table 38. Mapping of LDAP attribute names to RACF fixed fields (user) (continued)

RACF segment name	RACF keyword in altuser/adduser/listuser	LDAP attribute name
User OVM segment	PROGRAM	racfOvmInitialProgram
User OVM segment	UID	racfOvmUid
PROXY segment	BINDDN	racfLDAPBindDN
PROXY segment	BINDPW - value is not displayed by LDAP	racfLDAPBindPw
PROXY segment	LDAPHOST	racfLDAPHost
TSO segment	ACCTNUM	SAFAccountNumber
TSO segment	COMMAND	SAFDefaultCommand
TSO segment	DEST	SAFDestination
TSO segment	HOLDCLASS	SAFHoldClass
TSO segment	JOBCLASS	SAFJobClass
TSO segment	MAXSIZE	SAFMaximumRegionSize
TSO segment	MSGCLASS	SAFMessageClass
TSO segment	PROC	SAFDefaultLoginProc
TSO segment	SECLABEL	SAFTsoSecurityLabel
TSO segment	SIZE	SAFLogonSize
TSO segment	SYSOUTCLASS	SAFDefaultSysoutClass
TSO segment	UNIT	SAFDefaultUnit
TSO segment	USERDATA	SAFUserdata
WORKATTR segment	WAACCNT	racfWorkAttrAccountNumber
WORKATTR segment	WAADDR1	racfAddressLine1
WORKATTR segment	WAADDR2	racfAddressLine2
WORKATTR segment	WAADDR3	racfAddressLine3
WORKATTR segment	WAADDR4	racfAddressLine4
WORKATTR segment	WABLDG	racfBuilding
WORKATTR segment	WADEPT	racfDepartment
WORKATTR segment	WANAME	racfWorkAttrUserName
WORKATTR segment	WAROOM	racfRoom

Table 39. Mapping of LDAP attribute names to RACF fixed fields (group)

RACF segment name	RACF keyword in altgroup/addgroup/listgrp	LDAP attribute name
Group base	SUPGROUP	racfSuperiorGroup
Group base	Not modifiable - displayed as SUBGROUP(S)	racfSubGroupName
Group base	TERMUACC	racfGroupNoTermUAC
Group base	UNIVERSAL	racfGroupUniversal
Group base	Not modifiable - displayed as USER(S)	racfGroupUserids
User base or Group base	Not modifiable - displayed as CREATED	racfAuthorizationDate

Table 39. Mapping of LDAP attribute names to RACF fixed fields (group) (continued)

RACF segment name	RACF keyword in altgroup/addgroup/listgrp	LDAP attribute name
User base or Group base	DATA	racfInstallationData
User base or Group base	MODEL	racfDatasetModel
User base or Group base	OWNER	racfOwner
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
Group OMVS segment	GID	racfOmvsGroupId
Group OMVS segment	SHARED, AUTOGID	racfOmvsGroupIdKeyword
Group OVM segment	GID	racfOvmGroupId

Table 40. Mapping of LDAP attribute names to RACF fixed fields (connection)

RACF segment name	RACF keyword in connect	LDAP 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

Table 41. Mapping of LDAP attribute names to RACF fixed fields (resource)

RACF segment name	RACF keyword in rdefine/ralter/permit	LDAP attribute name
Resource base	Multi-value: SINGLEDSN, TVTOC, WARNING, or any other one-word values, such as NOKERB	racfResourceAttributes
Resource base	ADDCATEGORY	racfSecurityCategoryList
Resource base	ADDMEM	racfMemberList
Resource base	ADDVOL	racfVolumeList
Resource base	Not modifiable - displayed as ALTER COUNT	racfAlterAccessCount
Resource base	APPLDATA	racfApplData
Resource base	AUDIT	racfResourceAudit
Resource base	Not modifiable - displayed as AUTOMATIC	racfAutomatic
Resource base	Not modifiable - displayed as CONTROL COUNT	racfControlAccessCount

Table 41. Mapping of LDAP attribute names to RACF fixed fields (resource) (continued)

RACF segment name	RACF keyword in rdefine/ralter/permit	LDAP attribute name
Resource base	Not modifiable - displayed as CREATION DATE	racfAuthorizationDate
Resource base	DATA	racfInstallationData
Resource base	FCLASS, FGENERIC, FROM, FVOLUME - value is not displayed by LDAP	racfCopyProfileFrom
Resource base	GLOBALAUDIT	racfResourceGlobalAudit
Resource base	Not modifiable - displayed as LAST CHANGE DATE	racfLastReferenceDate
Resource base	LEVEL	racfLevel
Resource base	NOTIFY	racfNotify
Resource base	OWNER	racfOwner
Resource base	Not modifiable - displayed as READ COUNT	racfReadAccessCount
Resource base	SECLABEL	racfSecurityLabel
Resource base	SECLEVEL	racfSecurityLevel
Resource base	TIMEZONE	racfTimeZone
Resource base	UACC	racfUacc
Resource base	Not modifiable - displayed as UPDATE COUNT	racfUpdateAccessCount
Resource base	WHEN(DAYS())	racfLogonDays
Resource base	WHEN(TIME())	racfLogonTime
Resource base	Any of these PERMIT command keywords: ACCESS, DELETE, FCLASS, FGENERIC, FROM, FVOLUME, ID, RESET, WHEN	racfAccessControl
CDTINFO segment	CASE	racfCdtinfoCase
CDTINFO segment	DEFAULTRC	racfCdtinfoDefaultRc
CDTINFO segment	DEFAULTUACC	racfCdtinfoDefaultUacc
CDTINFO segment	FIRST	racfCdtinfoFirst
CDTINFO segment	GENERIC	racfCdtinfoGeneric
CDTINFO segment	GENLIST	racfCdtinfoGenList
CDTINFO segment	GROUP	racfCdtinfoGroup
CDTINFO segment	KEYQUALIFIERS	racfCdtinfoKeyQualifiers
CDTINFO segment	MACPROCESSING	racfCdtinfoMacProcessing
CDTINFO segment	MAXLENGTH	racfCdtinfoMaxLength
CDTINFO segment	MAXLENX	racfCdtinfoMaxLengthX
CDTINFO segment	MEMBER	racfCdtinfoMember
CDTINFO segment	OPERATIONS	racfCdtinfoOperations
CDTINFO segment	OTHER	racfCdtinfoOther
CDTINFO segment	POSIT	racfCdtinfoPosit
CDTINFO segment	PROFILESALLOWED	racfCdtinfoProfilesAllowed
CDTINFO segment	RACLIST	racfCdtinfoRacList
CDTINFO segment	SECLABELSREQUIRED	racfCdtinfoSecLabelsRequired

Table 41. Mapping of LDAP attribute names to RACF fixed fields (resource) (continued)

RACF segment name	RACF keyword in rdefine/ralter/permit	LDAP attribute name
CDTINFO segment	SIGNAL	racfCdtinfoSignal
CFDEF segment	FIRST	racfCfdefFirst
CFDEF segment	HELP	racfCfdefHelp
CFDEF segment	LISTHEAD	racfCfdefListHead
CFDEF segment	MAXLENGTH	racfCfdefMaxLength
CFDEF segment	MAXVALUE	racfCfdefMaxValue
CFDEF segment	MINVALUE	racfCfdefMinValue
CFDEF segment	MIXED	racfCfdefMixed
CFDEF segment	OTHER	racfCfdefOther
CFDEF segment	TYPE	racfCfdefType
DLFDATA segment	JOBNAMES	racfDlfdataJobNames
DLFDATA segment	RETAIN	racfDlfdataRetain
EIM segment	DOMAINDN	racfEimDomainDn
EIM segment	KERBREGISTRY	racfEimKerbRegistry
EIM segment	LOCALREGISTRY	racfEimLocalRegistry
EIM segment	OPTIONS	racfEimOptions
EIM segment	X509REGISTRY	racfEimX509Registry
ICSF segment	ASYMUSAGE	racfIcsfAsymUsage
ICSF segment	SYMEXPORTABLE	racfIcsfSymExportable
ICSF segment	SYMEXPORTCERTS	racfIcsfSymExportCerts
ICSF segment	SYMEXPORTKEYS	racfIcsfSymExportKeys
ICTX segment	DOMAP	racfIctxDoMap
ICTX segment	MAPPINGTIMEOUT	racfIctxMappingTimeOut
ICTX segment	MAPREQUIRED	racfIctxMapRequired
ICTX segment	USEMAP	racfIctxUseMap
KERB segment	DEFTKTLFE	racfKerbDefaultTicketLife
KERB segment	ENCRYPT	racfEncryptType
KERB segment	KERBNAME	krbPrincipalName
KERB segment	Not modifiable - displayed as KEY VERSION	racfCurKeyVersion
KERB segment	MAXTKTLFE	maxTicketAge
KERB segment	MINTKTLFE	racfKerbMinTicketLife
KERB segment	PASSWORD - value is not displayed by LDAP	racfKerbPassword
PROXY segment	BINDDN	racfLDAPBindDn
PROXY segment	BINDPW - value is not displayed by LDAP	racfLDAPBindPw
PROXY segment	LDAPHOST	racfLDAPHost
SESSION segment	CONVSEC	racfSessionConvSec
SESSION segment	INTERVAL	racfSessionInterval
SESSION segment	LOCK	racfSessionLock

Table 41. Mapping of LDAP attribute names to RACF fixed fields (resource) (continued)

RACF segment name	RACF keyword in rdefine/ralter/permit	LDAP attribute name
SESSION segment	SESSKEY	racfSessionSessKey
SIGVER segment	FAILLOAD	racfSigverFailLoad
SIGVER segment	SIGAUDIT	racfSigverSigAudit
SIGVER segment	SIGREQUIRED	racfSigverSigRequired
SSIGNON segment	KEYENCRYPTED - value is not displayed by LDAP	racfSsignonKeyEncrypted
SSIGNON segment	KEYMASKED - value is not displayed by LDAP	racfSsignonKeyMasked
STDATA segment	GROUP	racfStdataGroup
STDATA segment	PRIVILEGED	racfStdataPrivileged
STDATA segment	TRACE	racfStdataTrace
STDATA segment	TRUSTED	racfStdataTrusted
STDATA segment	USER	racfStdataUser

Table 42. Mapping of LDAP attribute names to RACF fixed fields (setropts)

RACF segment name	RACF keyword in setropts	LDAP attribute name
Setropts base	Multi-value: REFRESH, WHEN(PROGRAM)	racfSetroptsAttributes
Setropts base	AUDIT	racfAudit
Setropts base	CLASSACT	racfClassAct
Setropts base	GENCMD	racfGenCmd
Setropts base	GENERIC	racfGeneric
Setropts base	GENLIST	racfGenList
Setropts base	GLOBAL	racfGlobal
Setropts base	LOGOPTIONS(ALWAYS)	racfLogOptionsAlways
Setropts base	LOGOPTIONS(DEFAULT)	racfLogOptionsDefault
Setropts base	LOGOPTIONS(FAILURES)	racfLogOptionsFailures
Setropts base	LOGOPTIONS(NEVER)	racfLogOptionsNever
Setropts base	LOGOPTIONS(SUCCESSES)	racfLogOptionsSuccesses
Setropts base	RACLIST	racfRacList
Setropts base	STATISTICS	racfStatistics

Note: The SDBM attribute types that return DN-type values are: **racfNotify, racfGroupUserids, racfConnectGroupName, racfDefaultGroup, racfSubGroupName, racfSuperiorGroup, racfConnectOwner, racfOwner, racfStdataUser, racfStdataGroup, racfAudit, racfCdtInfoGroup, racfCdtInfoMember, racfClassAct, racfGenCmd, racfGeneric, racfGenList, racfGlobal, racfLogoptionsAlways, racfLogoptionsDefault, racfLogoptionsFailures, racfLogoptionsNever, racfLogoptionsSuccesses, racfRacList**, and **racfStatistics**.

By default on search responses, these SDBM DN-type attribute values are returned in uppercase format (for example, `RACFID=X,PROFILETYPE=USER,CN=SDBM`, `RACFID=Y,PROFILETYPE=GROUP,CN=SDBM`, or `PROFILENAME=TMP,PROFILETYPE=FACILITY,CN=SDBM`).

However, if the lowest order bit for the decimal bitmask specified in the **LDAP_COMPAT_FLAGS** environment variable is set to 1 (for example, 1, 3, or 5), these SDBM DN-type attribute values are returned in mixed case format (for example, `racfid=X,profiletype=USER,cn=sdbm`, `racfid=Y,profiletype=GROUP,cn=sdbm`, or `profilename=TMP,profiletype=FACILITY,cn=sdbm`). See "LDAP_COMPAT_FLAGS environment variable" on page 208 for more information.

Associating LDAP attributes to RACF custom fields

The user and group profile custom fields in the CSDATA segment are not predefined by RACF, but are defined in RACF by the user. If those fields are to be set and displayed using LDAP, then the user must add an attribute to the LDAP schema to represent each custom field. The attribute is defined in the LDAP schema using an **attributeTypes** value and an **IBMAttributeTypes** value. LDAP does not allow more than one attribute to be associated with the same RACF custom field.

- The **attributeTypes** value specifies the object identifier (OID), name, syntax, and equality rule of the attribute. The OID and name must not be in use in the schema. The attribute name can be the same as the RACF custom field name, or it can be different. For example, if the custom field name is `phone`, the attribute name could be `phone`, or `workphone`, or anything else that is not already in use. See "Attribute types" on page 293 for more information about **attributeTypes**.
- The **IBMAttributeTypes** value must include the **RACFFIELD** keyword to identify the RACF custom field associated with the attribute. The **RACFFIELD** value specifies the resource profile name that is used to define the custom field in RACF, with periods (.) changed to dashes (-). For example, if the RACF custom field is defined by the `USER.CSDATA.PHONE` resource profile, **RACFFIELD** contains `USER-CSDATA-PHONE`. **RACFFIELD** also optionally specifies the type of RACF custom field. The accepted values are **char**, **flag**, **hex**, **num**, and **qchar**. If the RACF custom field is defined with `TYPE(CHAR) FIRST(ANY) OTHER(ANY)`, then specify **qchar** in **RACFFIELD** to indicate that SDBM should put the attribute value in quotations when creating RACF commands. Otherwise, specify in **RACFFIELD** the same value as was used for `TYPE` when defining the custom field in RACF. If a type value is not specified in **RACFFIELD**, LDAP assumes that the custom field type is **char**. See "Schema introduction" on page 276 for more information about **IBMAttributeTypes**.

For example, if the `phone` custom field is defined in the RACF user profile with `TYPE(CHAR)`, the following attribute could be added to the LDAP schema to represent the custom field:

```
attributetypes: (
      phone-OID
      NAME 'phone'
      DESC 'Represents the PHONE field in the RACF user CSDATA segment'
      EQUALITY caseIgnoreMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
      SINGLE-VALUE
      USAGE userApplications
      )
ibmattributetypes: (
      phone-OID
      ACCESS-CLASS sensitive
      RACFFIELD ('USER-CSDATA-PHONE' 'char')
      )
```

Note:

1. A numeric OID can be used instead of the nonnumeric OID `phone-OID`. A numeric OID must be used if the LDAP server is sharing a TDBM database with an Integrated Security Services LDAP server on z/OS V1R10 or earlier releases.
2. If the RACF custom field is defined to be case-sensitive (using `MIXED(YES)`), change the EQUALITY rule to `EQUALITY caseExactMatch`. Otherwise, compare operations can fail if mixed case values are involved.
3. The SYNTAX must be IA5 String (`1.3.6.1.4.1.1466.115.121.1.26`).
4. All RACF custom fields have only a single value, therefore, `SINGLE-VALUE` is specified.
5. The ACCESS-CLASS is 'sensitive' for most RACF attributes, but can be changed to 'critical' if the field contains data to which access is more restrictive. SDBM does not use the ACCESS-CLASS value, but TDBM, LDBM, and CDBM do.

For completeness, add an object class to the LDAP schema to represent the CSDATA segment in each profile. SDBM always assumes that the object class names are **racfUserCsdataSegment** for the CSDATA segment in the user profile or **racfGroupCsdataSegment** for the CSDATA segment in the group profile. SDBM adds this object class to a user or group entry if the corresponding RACF profile contains the CSDATA segment.

For example, if the `PHONE` and `SSN` custom fields are defined in RACF for the user profile and the LDAP attributes `phone` and `socialSecurityNumber` are defined in the LDAP schema to represent the custom fields, the following object class should be added to the LDAP schema:

```
objectclasses: (
      racfUserCsdataSegment-OID
      NAME 'racfUserCsdataSegment'
      DESC 'Represents the CSDATA segment in a z/OS RACF USER profile'
      SUP top
      AUXILIARY
      MAY ( phone $ socialSecurityNumber )
      )
```

Note: A numeric OID can be used instead of the nonnumeric OID `racfUserCsdataSegment-OID`. A numeric OID must be used if the LDAP server is sharing a TDBM database with an Integrated Security Services LDAP server on z/OS V1R10 or earlier releases.

Special usage of racfAttributes, racfConnectAttributes, racfResourceAttributes, and racfSetroptsAttributes

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'. Additional values, such as PASSWORD, can be returned in **racfAttributes** that are not returned by the **listuser** command.

Similarly, **racfConnectAttributes** can be used to specify any single-word keywords that are valid on a RACF **connect** command, as can **racfResourceAttributes** for the RACF **rdefine** and **ralter** commands. **racfSetroptsAttributes** can be used for the RACF **setropts** command, but only those values listed in Table 42 on page 337 can be specified.

RACF namespace entries

When the SDBM backend is used to make RACF information accessible over the LDAP protocol, SDBM creates a set of top entries to set up a hierarchical representation of RACF users, groups, connections, classes, resources, and class options. These top entries consist of the suffix, top user entry, top group entry, top connection entry, a top entry for each RACF class (except DATASET, which is not supported), and a setropts entry. For example, the top entries in Figure 24 and Figure 25 on page 341 are:

- `cn=RACFA,o=IBM,c=US` (suffix entry)
- `profileType=User,cn=RACFA,o=IBM,c=US` (top user entry)
- `profileType=Group,cn=RACFA,o=IBM,c=US` (top group entry)
- `profileType=Connect,cn=RACFA,o=IBM,c=US` (top connect entry)
- `profileType=Cdt,cn=RACFA,o=IBM,c=US` (top cdt class entry)
- `profileType=Facility,cn=RACFA,o=IBM,c=US` (top facility class entry)
- `cn=Setropts,cn=RACFA,o=IBM,c=US` (setropts entry)

The top entries cannot be added or deleted. Except for the setropts entry, the top entries can only be compared and searched.

The setropts entry can be modified, compared, and searched.

The value used for the suffix entry DN is the value specified for the **suffix** option in the SDBM section of the LDAP server configuration file (see "Setting up for SDBM" on page 60).).

Following is a high-level diagram of the RACF backend.

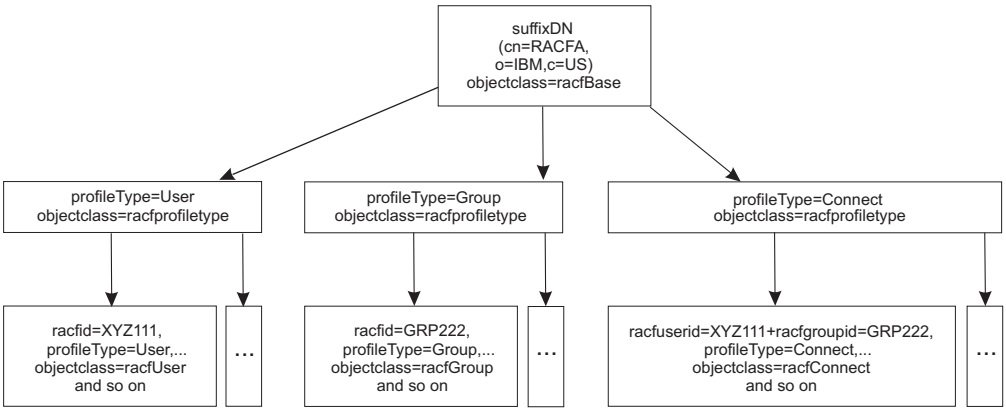


Figure 24. RACF namespace hierarchy (Part 1 of 2)

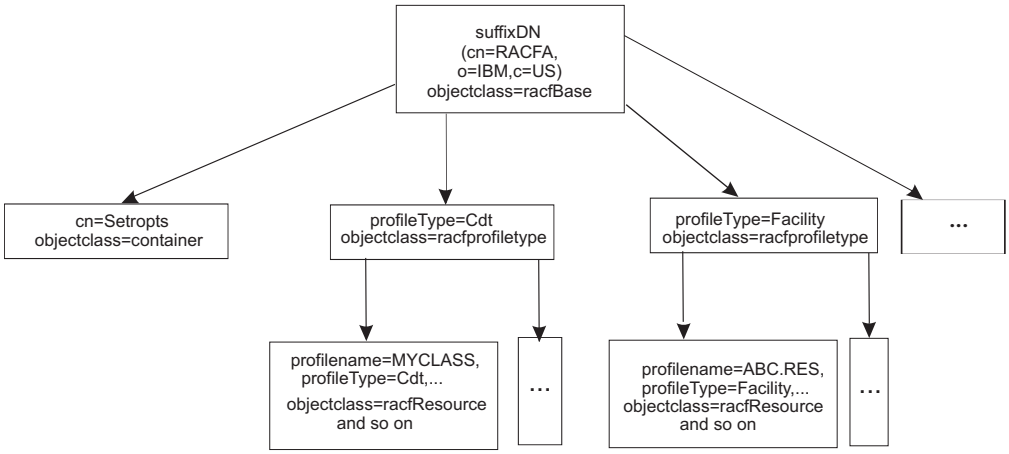


Figure 25. RACF namespace hierarchy (Part 2 of 2)

SDBM schema information

The attributes and object classes used by SDBM to represent RACF values are always in the LDAP server schema, except for any attributes needed for RACF custom fields.

SDBM support for special characters

An SDBM DN, including the SDBM suffix, can contain the following special characters:

- A plus sign (+), double quotation mark ("), or backslash (\) anywhere in a DN.
- A number sign (#) at the beginning of a value in a DN.

When present in a DN, a special character must be escaped by preceding it with a single backslash (\). Note that the suffix in the LDAP server configuration file must use two back slashes (\\) to escape a special character, but only a single backslash is used in a DN.

For example, if the SDBM suffix in the configuration file is

```
suffix cn=\\#plex#1
```

then the DN for the RACF resource profile `a+b` in class `#x#y` would be

```
profilename=a\+b,profiletype=\#x#y,cn=\#plex#1
```

Special characters in a DN returned by SDBM are always escaped by a single backslash. Number signs that are not at the beginning of a value and equal signs (=) might be escaped, depending on the usage of the DN.

When specifying a value containing a special character for an attribute within an add or modify request, escape the special character with a back slash if the attribute is part of a DN, otherwise, do not escape the special character. For instance, to add a user with the default group `#d1grp`, specify either:

```
racfdefaultgroup: racfid=\#d1grp,profiletype=group,cn=\#plex#1
```

or

```
racfdefaultgroup: #d1grp
```

within the entry.

When specifying a value containing a special character for an attribute within a search filter, the special character can be escaped or not. For instance, to search for all RACF users starting with `#user`, use the search filter `racfid=#user*` or `racfid=\#user*`.

Control of access to RACF data

As explained above, SDBM operations result in issuing RACF commands. Table 43 and Table 44 on page 350 indicate which commands are issued for various SDBM operations. The RACF commands are issued under the context of the RACF user ID that is bound to SDBM. RACF determines the results of the RACF commands based on the RACF authority of that user ID. If the RACF command fails, the SDBM operation fails and returns any error information issued by RACF.

In particular, the RACF **search** command can fail because of lack of authority, even if the bound user is able to extract RACF data from user IDs that match the RACF **search**. In this case, SDBM search operations that issue a RACF **search** command can fail and return the following:

```
ldap_search: Unknown error
ldap_search: additional info: ICH31005I NO ENTRIES MEET SEARCH CRITERIA
```

SDBM operational behavior

Table 43 shows how SDBM behaves during different LDAP operations.

Table 43. 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 356 **Bind** Error: No credentials

Table 43. RACF backend behavior (continued)

Target DN	LDAP operation behavior
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 356 **Search subtree** See "Searching the entire RACF database" on page 356 **Bind** Error: No credentials
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 356 **Search subtree** See "Searching the entire RACF database" on page 356 **Bind** Error: No credentials

Table 43. RACF backend behavior (continued)

Target DN	LDAP operation behavior
profiletype=Facility,*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 356 **Search subtree** See "Searching the entire RACF database" on page 356 **Bind** Error: No credentials
cn=setropts,*suffixDN*	**Add** Error: Unwilling to perform **Modify** Perform a **setropts** RACF command **Delete** Error: Unwilling to perform **Modify DN** Error: Unwilling to perform **Compare** Compare attribute **Search base** Perform an R_admin **setropts extract** RACF command **Search one level** Empty search results (this is a leaf node in the hierarchy) **Search subtree** Perform an R_admin **setropts extract** RACF command **Bind** Error: No credentials

Table 43. RACF backend behavior (continued)

Target DN	LDAP operation behavior
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 356 **Search subtree** See "Searching the entire RACF database" on page 356 **Bind** Error: No credentials
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 an R_admin profile extract RACF command using USER=XYZ111 **Search base** Perform an R_admin profile extract RACF command using USER=XYZ111 **Search one level** Empty search results (this is a leaf node in the hierarchy) **Search subtree** Perform an R_admin profile extract RACF command using USER=XYZ111 **Bind** If bind type is not simple, Error: Unwilling to perform or else use the **RACROUTE REQUEST=VERIFY, ENVIR=CREATE** macro to verify the user ID and password or password phrase combination. The RACF groups the user belongs to are determined on the next LDAP operation.

Table 43. RACF backend behavior (continued)

Target DN	LDAP operation behavior
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 an R_admin profile extract RACF command using GROUP=GRP222 **Search base** Perform an R_admin profile extract RACF command using GROUP=GRP222 **Search one level** Empty search results (this is a leaf node in the hierarchy) **Search subtree** Perform an R_admin profile extract RACF command using GROUP=GRP222 **Bind** Error: No credentials
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 an R_admin profile extract RACF command using USER=XYZ111 **Search base** Perform an R_admin profile extract RACF command using USER=XYZ111 **Search one level** Empty search results (this is a leaf node in the hierarchy) **Search subtree** Perform an R_admin profile extract RACF command using USER=XYZ111 **Bind** Error: No credentials

Table 43. RACF backend behavior (continued)

Target DN	LDAP operation behavior	
profilename=ABC.RES,profiletype=Facility, *suffixDN*	**Add**	Perform an **rdefine** and, possibly, **permit** RACF commands for PROFILE=ABC.RES in CLASS=FACILITY. A separate **permit** command is performed for each **racfAccessControl** attribute value that is specified.
	Modify	Perform an **ralter** or **permit**, or both RACF commands for PROFILE=ABC.RES in CLASS=FACILITY. A separate **permit** command is performed for each **racfAccessControl** attribute value that is specified. An **ralter** command is performed if an attribute other than **racfAccessControl** is specified.
	Delete	Perform an **rdelete** RACF command for PROFILE=ABC.RES in CLASS=FACILITY
	Modify DN	Error: Unwilling to perform
	Compare	Compare requested attribute with data returned from an R_admin profile extract RACF command using PROFILE=ABC.RES in CLASS=FACILITY
	Search base	Perform an R_admin profile extract RACF command using PROFILE=ABC.RES in CLASS=FACILITY
	Search one level	Empty search results (this is a leaf node in the hierarchy)
	Search subtree	Perform an R_admin profile extract RACF command using PROFILE=ABC.RES in CLASS=FACILITY
	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 completed a partial update to an entry in RACF. The results match what occurs if the update is done by using the RACF **altuser, altgroup, connect, ralter**, and **permit** 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.

This is further complicated when adding or modifying a general resource profile because this can involve multiple RACF commands: a **rdefine** or **ralter** command followed by one or more **permit** commands. If one of the commands fails, processing ends but the resource profile is still updated with the results of the prior successful commands.

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 from normal LDAP support:

- When adding a connection entry that exists, the entry is modified using the specified attributes. There is no indication returned that the entry 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. However, the entire attribute value is processed as case-sensitive in a compare operation. 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, group, or class 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.
4. For multi-value attributes, the RACF **altuser** and **ralter** commands do 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, racfAudit, racfClassAct, racfClassName, racfConnectAttributes, racfGenCmd, racfGeneric, racfGenList, racfGlobal, racfLevelKeyword, racfLogonDays, racfLogOptionsAlways, racfLogOptionsDefault, racfLogOptionsFailures, racfLogOptionsNever, racfLogOptionsSuccessess, racfMemberList, racfMformKeyword, racfMonitorKeyword, racfRacList, racfResourceAttributes, racfSecurityCategoryList, racfSetroptsAttributes, racfStatistics, racfVolumeList**.

- Values for the following multi-value attributes always replace the existing value (even if add is specified): **racfCdtinfoFirst, racfCdtinfoOther, racfDlfdataJobNames, racfDomains, racfIcsfSymExportCerts, racfIcsfSymExportKeys, racfMscopeSystems, racfNetviewOperatorClass, racfOperatorClass, racfResourceAudit, racfResourceGlobalAudit, racfRoutcodeKeyword, racfRslKey, racfTslKey**.
- 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, racfAccessControl**, and **racfIcsfAsymUsage**.

For single-value attributes, there is no difference between using an add modification or a replace modification to set the value. For either type of modification, the value is added if the attribute value does not exist and the value replaces the existing attribute value, if there is one.

5.

6. In order to update attributes related to CICS, CICS must be set up on your system; otherwise, errors result.
7. For modify, if a request is made to delete a specific attribute value for an attribute where specific values cannot be selectively deleted, an **LDAP_UNWILLING_TO_PERFORM** error code is returned. Similarly, if a request is made to delete the entire attribute for an attribute where specific values to delete must be specified, an **LDAP_UNWILLING_TO_PERFORM** error code is returned.

   The following attributes require that specific values deleted be specified: **racfAudit, racfClassAct, racfClassName, racfGenCmd, racfGeneric, racfGenList, racfGlobal, racfMemberList, racfRacList, racfStatistics, racfVolumeList**.

   The following attributes allow specifying specific values to delete but also support deleting the entire attribute: **racfAccessControl, racfAttributes, racfConnectAttributes, racfResourceAttributes, racfSecurityCategoryList, racfSetroptsAttributes**.

   All other attributes that have a delete command in RACF only allow deleting the entire attribute. If an attempt is made to delete any attribute that has no corresponding delete command in RACF, an **LDAP_UNWILLING_TO_PERFORM** error code is returned.
8. The **racfCopyProfileFrom** attribute is used to specify any combination of the RACF **rdefine** FCLASS, FGENERIC, FROM, and FVOLUME keywords and values to indicate a resource profile to use as a model when creating a new resource profile. The value specified for this attribute must be syntactically correct for an **rdefine** command and is inserted as is in the command. For example, the following uses the RES.MODEL resource profile in the FACILITY class as a model:

   ```
   racfcopyprofilefrom: FROM(RES.MODEL) FCLASS(FACILITY)
   ```
9. The **racfAccessControl** attribute is used to manage the access control lists for a general resource profile. Each attribute value is used to create a separate RACF **permit** command. Each value must be a syntactically correct RACF **permit** command, without the class and profile names. SDBM adds the class and profile names before issuing the RACF **permit** command. It also adds the DELETE keyword for a modify request to delete the value.

   Note: When issuing multiple RACF **permit** commands for the same resource, the order of the **permit** commands can be critical. SDBM issues a **permit** command for each **racfAccessControl** value in the order that SDBM receives

the values. If you specify multiple add, replace, and delete changes to an attribute in a single modify operation, many **ldapmodify** utilities (including the z/OS client **ldapmodify** utility) might reorder the changes to put all the changes of the same type together. Therefore, the values as presented to SDBM might not be in the original order and the results of the **permit** commands might not be as you want. To avoid this, separate different **racfAccessControl** attribute changes into separate modify operations.

When LDAP returns the **racfAccessControl** value during a search operation, the value might contain the COUNT field if this is part of the RACF output, for example:

```
racfaccesscontrol: ID(X) ACCESS(READ) COUNT(5)
```

If SDBM finds the COUNT field in a **racfAccessControl** value during an add or modify operation, the field is removed from the value before the value is used to generate a RACF **permit** command. This allows LDAP search output to be used as add or modify input.

When using the **racfAccessControl** attribute in a compare operation, the comparison is done only on the value specified for the ID keyword within the attribute value. The rest of the attribute value is not used. If the ID value is contained in any access control list within the resource profile, the compare returns **LDAP_COMPARE_TRUE**. If the attribute value does not have the ID keyword, has more than one ID value, or the value is not contained in any access list within the resource profile, compare returns **LDAP_COMPARE_FALSE**. Basically, a **racfAccessControl** compare operation can be used to determine if a specific RACF user or group appears in the access control lists within a resource profile.

SDBM search capabilities

SDBM supports a limited set of search filters. The following table describes each supported filter and indicates from what bases it is valid, what type of entries it returns (a complete entry or entries that contain the DN of the entry), and what RACF commands are issued to perform the search. Most searches can only be performed from one of these top entries: the *suffix* entry, the `profiletype=user,`*suffix* entry, the `profiletype=group,`*suffix* entry, the `profiletype=connect,`*suffix* entry, and the `profiletype=`*class*`,`*suffix* entries.

Table 44. SDBM search filters

Filter	Search behavior
krbprincipalname=*any_value*	**Description:** find user profile for the RACF user whose KERB KERBNAME value is *any_value* **Allowed base:** *suffix* `profiletype=user,`*suffix* **Returns:** complete entry **Commands:** – **R_usermap** – followed by **R_admin user profile extract**

Table 44. SDBM search filters (continued)

Filter	Search behavior
objectclass=*	**Description:** match any user, group, connection, resource profile, and setropts **Allowed base:** any SDBM entry **Returns:** • DN-only entries if scope includes all users, groups, connections, resource profiles, or setropts • Complete entry if scope includes a single entry **Commands:** • if scope includes all users: **search class(user) filter**(*) • if scope includes all groups: **search class(group) filter**(*) • if scope includes all connections: – **search class(group) filter**(*) – followed by **R_admin group profile extract** for each group • if scope includes all classes: – **RACROUTE STAT** to retrieve all class names – followed by **search class***(className)* **filter**(**) for each class • if scope includes a specific class: – **RACROUTE STAT** to determine if the class exists – followed by **search class***(className)* **filter**(**) for the class • if scope includes a single user: **R_admin user profile extract** • if scope includes a single group: **R_admin group profile extract** • if scope includes a single connection: **R_admin connect profile extract** • if scope includes a single resource: **R_admin resource profile extract** • if scope includes just the cn=setropts entry: **R_admin setropts extract**

Table 44. SDBM search filters (continued)

Filter	Search behavior
profilename=*any_value*	**Description:** find the RACF general resource profiles whose names match *any_value* (can contain wildcards) **Note:** RACF profile names might be case-sensitive, depending on the class. **Allowed base:** *suffix* profiletype=*className*,*suffix* **Returns:** DN-only entries **Commands:** • if scope includes all classes: – **RACROUTE STAT** to retrieve all class names – followed by **search class***(className)* **filter***(any_value)* for each class • if scope includes a single class: – **RACROUTE STAT** to determine if the class exists – followed by **search class***(className)* **filter***(any_value)* for the class
racfgroupid=*any_value*	**Description:** find connection profiles for members of the RACF groups whose names match *any_value* (can contain wildcards) **Allowed base:** *suffix* profiletype=connect,*suffix* **Returns:** DN-only entries **Commands:** • if no wildcard in *any_value*: **R_admin group profile extract** • if wildcard in *any_value*: – **search class(group) filter**(*any_value*) – followed by **R_admin group profile extract** for each group

Table 44. SDBM search filters (continued)

Filter	Search behavior
racfid=*any_value*	**Description:** find user and group profiles for the RACF users and groups whose names match *any_value* (can contain wildcards) **Allowed base:** *suffix* `profiletype=user,`*suffix* `profiletype=group,`*suffix* **Returns:** DN-only entries **Commands:** • if scope includes all users: **search class(user) filter**(*any_value*) • if scope includes all groups: **search class(group) filter**(*any_value*)
racflnotesshortname=*any_value*	**Description:** find user profile for the RACF user whose LNOTES SNAME value is *any_value* **Allowed base:** *suffix* `profiletype=user,`*suffix* **Returns:** complete entry **Commands:** – **R_usermap** – followed by **R_admin user profile extract**
racfndsusername=*any_value*	**Description:** find user profile for the RACF user whose NDS UNAME value is *any_value* **Allowed base:** *suffix* `profiletype=user,`*suffix* **Returns:** complete entry **Commands:** – **R_usermap** – followed by **R_admin user profile extract**

Table 44. SDBM search filters (continued)

Filter	Search behavior
racfomvsgroupid=*number*	**Description:** find group profile for one of the RACF groups whose OMVS GID values match *number* **Allowed base:** *suffix* `profiletype=group,`*suffix* **Returns:** complete entry **Commands:** – **getgrgid**(*number*) – followed by **R_admin group profile extract**
racfomvsgroupid;allOMVSids=*number*	**Description:** find group profiles for all the RACF groups whose OMVS GID values match *number* **Allowed base:** *suffix* `profiletype=group,`*suffix* **Returns:** DN-only entries **Commands:** **search class(group) gid**(*number*)
racfomvsuid=*number*	**Description:** find user profile for one of the RACF users whose OMVS UID values match *number* **Allowed base:** *suffix* `profiletype=user,`*suffix* **Returns:** complete entry **Commands:** – **getpwuid**(*number*) – followed by **R_admin user profile extract**
racfomvsuid; allOMVSids=*number*	**Description:** find user profiles for all the RACF users whose OMVS UID values match *number* **Allowed base:** *suffix* `profiletype=user,`*suffix* **Returns:** DN-only entries **Commands:** **search class(user) uid**(*number*)

Table 44. SDBM search filters (continued)

Filter	Search behavior
racfuserid=*any_value*	**Description:** find connection profiles for RACF users whose names match *any_value* (can contain wildcards) **Allowed base:** *suffix* `profiletype=connect,`*suffix* **Returns:** DN-only entries **Commands:** • if no wildcard in *any_value*: **R_admin user profile extract** • if wildcard in *any_value* – **search class(user) filter**(*any_value*) – followed by **R_admin user profile extract** for each user
(&(racfuserid=*any_value1*) (racfgroupid=*any_value2*))	**Description:** find connection profiles for RACF users whose names match *any_value*1 and who belong to RACF groups whose names match *any_value2* (both can contain wildcards) **Allowed base:** *suffix* `profiletype=connect,`*suffix* **Returns:** DN-only entries **Commands:** • if no wildcard in *any_value1*: **R_admin user profile extract** • if no wildcard in *any_value2* **R_admin group profile extract** • if wildcard in both *any_value1* and *any_value2* – **search class(group) filter**(*any_value2*) – followed by **R_admin group profile extract** for each group

Except for the AND filter for connections, complex search filters that include NOT, AND, OR, LE, or GE constructs are not supported.

The values for the **profilename, racfgroupid, racfid**, and **racfuserid** filters can include the wildcards supported by RACF. These wildcards 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`.

To include multiple levels of qualifiers in a resource profile name search, include either ** or ** in the **profilename** filter. For example, profilename=XYZ.** searches for all resource profiles that have XYZ as the first qualifier. Do not use ** in the filter because this is not a valid LDAP filter. The result of a search with the filter profilename=** is:

```
ldap_search: Protocol error
ldap_search: additional info: R010043 Substring filter for attribute 'profilename' has no value
```

Although an '*' or '**' can be part of a resource profile name, there is no way to indicate in the **profilename** filter that an asterisk or double asterisk is part of the name rather than a wildcard. For example, a search using a filter such as profilename=ABC* returns all profile names beginning with ABC, including the ABC* profile (if it exists).

Note about searching universal groups: Most of the members of a RACF universal group are not contained in the list of members of the group. As a result, a search of the entry for a universal group does not return most of the members of the group. 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 wildcard, then the connection entry is returned for the specified **racfuserid**.
- If the **racfuserid** part of the connection search filter contains a wildcard, then the connection entry for a user is returned only if the user is explicitly contained in the list of members of the universal group.

Searching the entire RACF database

Most searches that query the entire RACF database, for example, a subtree search from any of the top directory entries except the setopts entry, return only the DN (distinguished name) attribute. You may then obtain more specific data about a particular user, group, connection, or resource on a follow-up search using a specific DN as the search base.

The exceptions to this are searches using the “application ID” filters:

```
krbprincipalname=<any_name>
racflnotesshortname=<any_value>
racfndsusername=<any_value>
racfomvsgroupid=<number>
racfomvsuid=<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 certain LDAP search requests, SDBM uses the RACF **R_admin** "run command" interface to issue RACF **search** commands. The **R_admin** "run command" interface limits the number of records in its output to 4096. This means that the RACF **search** command output might be incomplete if you have many users, groups, connections, or resources. See *z/OS Security Server RACF Callable Services* on the RACF restriction. The restriction only affects those SDBM searches that issue the RACF **search** command. See Table 44 on page 350 to determine which SDBM searches are affected.

RACF restriction on amount of input: RACF limits the number of operands that are specified in RACF commands. If the number of operands surpasses this limit, RACF ignores some of the operands and processes the command. Therefore, an SDBM add or modify operation containing many attributes appears to run

successfully but some of the attributes might not be set. For more information, see *z/OS Security Server RACF Command Language Reference*.

LDAP restriction on RACF data: Except for the RACF user password or password phrase envelopes, all field values sent by RACF to LDAP must consist of printable characters. If a RACF field contains unprintable characters, the value returned in the LDAP output does not match the RACF value and is not printable. If a RACF field contains binary zeros, the LDAP output might be truncated. In particular, make sure that the installation DATA field in RACF user and resource profiles does not contain binary zeros or other unprintable characters.

Retrieving RACF user password and password phrase envelopes

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. Similarly, the RACF user password phrase envelope is returned when the **racfPassPhraseEnvelope** attribute is specified on the search. Each envelope is returned by the LDAP server as a binary data berval (binary data and length). If the **racfPasswordEnvelope** and **racfPassPhraseEnvelope** attributes are not specified on the search request, the RACF envelopes are not returned.

Note: When using a utility such as **ldapsearch** to retrieve the password or password phrase envelopes, the returned value is base-64 encoded.

Changing a user password or password phrase in RACF using SDBM

There are two ways to use SDBM to change a user password or password phrase in RACF.

1. The user password or password phrase of the bind user can be changed during an LDAP simple bind to SDBM. The simple bind occurs as part of an LDAP function such as search, add, modify, compare, or delete. The password or password phrase change is provided in the password portion of the LDAP simple bind. The change must be in the following format:

   ```
   currentvalue/newvalue
   ```

   The current and new value must both be passwords or password phrases. An error is returned if one of the values is a password and the other is a password phrase.

   The forward slash (/) is used as the indication of a password or password phrase change during the LDAP simple bind. Password or password phrase 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 or password phrase change fails with LDAP return code **LDAP_INVALID_CREDENTIALS** and LDAP reason code of:

   ```
   R000101  The new password is not valid
   ```

   if the new password or password phrase does not pass the rules established on the system.

   Note: A forward slash (/) is a legal character in a password phrase (but not in a password). During SDBM bind, a backward slash (\) is an escape character to indicate that the next character is part of the password or password phrase and has no special meaning. The backward slash is removed during bind processing. Therefore, during bind, a forward slash in a password phrase must

be preceded by a backward slash to indicate that the forward slash is part of the password phrase and is not the password phrase change indicator. For example, the password phrase `this1slash/ispartofthevalue2use` must be specified as `this1slash\/ispartofthevalue2use` during bind. A backward slash is also a legal character in a password phrase (but not in a password). Therefore, a backward slash in a password phrase must be preceded by another backward slash to indicate that it is not an escape character.

Once the bind succeeds, the password or password phrase is changed even if the LDAP function eventually fails.

For example, the following command changes the password phrase 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  -b "" objectclass=*
```

2. To change any RACF 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`.

   A RACF user's password phrase is changed the same way as described above, using the **racfPassPhrase** attribute.

Using LDAP client utilities with SDBM

The LDAP client utilities described in *z/OS IBM Tivoli Directory Server Client Programming for z/OS* 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: 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 adds 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 non-default data that RACF displays on a **listuser** command, but using LDAP attribute names. Following is an example for **newuser**:

```
racfid=NEWUSER,profiletype=USER,sysplex=SYSPLEXA
racfid=NEWUSER
racfauthorizationdate=07/18/05
racfowner=RACFID=ADMIN1,PROFILETYPE=USER,SYSPLEX=SYSPLEXA
racfpasswordinterval=186
racfdefaultgroup=RACFID=G1,PROFILETYPE=GROUP,SYSPLEX=SYSPLEXA
racflogondays=SUNDAY
racflogondays=MONDAY
racflogondays=TUESDAY
racflogondays=WEDNESDAY
racflogondays=THURSDAY
racflogondays=FRIDAY
racflogondays=SATURDAY
racflogontime=ANYTIME
racfconnectgroupname=RACFID=G1,PROFILETYPE=GROUP,SYSPLEX=SYSPLEXA
racfhavepasswordenvelope=YES
racfhavepassphraseenvelope=YES
racfpassphrasechangedate=06/11/07
racfattributes=PASSWORD
racfattributes=PASSPHRASE
safaccountnumber=123
safholdclass=H
saflogonsize=1024
safmaximumregionsize=0
safuserdata=0000
objectclass=RACFBASECOMMON
objectclass=RACFUSER
objectclass=SAFTSOSEGMENT
```

Example: searching for a user's password and password phrase envelopes in RACF

The following search returns the **racfPasswordEnvelope** and **racfPassPhraseEnvelope** attributes:

```
ldapsearch -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,sysplex=sysplexa
 -w passwd -L -b racfid=newuser,profiletype=user,sysplex=sysplexa
 "objectclass=*" racfpasswordenvelope racfpassphraseenvelope
```

The result returned is:

```
dn: racfid=newuser,profiletype=user,sysplex=sysplexa
racfpasswordenvelope:: base-64_encoded_password_envelope
racfpassphraseenvelope:: base-64_encoded_passphrase_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 LDAP commands for modifying, searching, and removing a RACF group using SDBM are similar to the corresponding commands for a RACF user. For more information, see the examples in this section for a RACF user.

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 non-default information from the GROUP section that RACF displays on a **listuser** command, but using LDAP attribute names. Following is an example for `newuser`'s connection to `grp222`:

```
racfuserid=NEWUSER+racfgroupid=GRP222,profiletype=CONNECT,sysplex=SYSPLEXA
racfuserid=NEWUSER
racfgroupid=GRP222
racfconnectauthdate=07/18/05
racfconnectowner=RACFID=ADMIN1,PROFILETYPE=USER,SYSPLEX=SYSPLEXA
racfconnectgroupauthority=USE
racfconnectgroupuacc=NONE
racfconnectcount=0
objectclass=RACFBASECOMMON
objectclass=RACFCONNECT
```

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=G1,profiletype=CONNECT,sysplex=sysplexa

racfuserid=NEWUSER+racfgroupid=GRP222,profiletype=CONNECT,sysplex=sysplexa
```

Note that G1 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"
```

Example: adding a resource profile in the facility class and giving a user and a group access to the profile

If the LDIF file `resource.add` contains:

```
dn: profilename=NEW.RESOURCE,profiletype=facility,sysplex=sysplexa
objectclass: racfResource
objectclass: extensibleObject
profilename: NEW.RESOURCE
racfuacc: read
racfnotify: admin1
racfaccesscontrol: ID(u1) ACCESS(UPDATE)
racfaccesscontrol: ID(g1) ACCESS(CONTROL) WHEN(TERMINAL(T2))
```

The following command adds the `NEW.RESOURCE` resource profile to the facility class in RACF and gives the requested access:

```
ldapadd -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,sysplex=sysplexa
 -w passwd -f resource.add
```

Note: A RACF **rdefine** command, followed by two RACF **permit** commands, are issued.

Example: refreshing the raclist for the facility class

If the LDIF file `setropts.mod` contains:

```
dn: cn=setropts,sysplex=sysplexa
changetype: modify
racfraclist: facility
racfsetroptsattributes: refresh
```

The command:

```
ldapmodify -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,sysplex=sysplexa -w passwd
 -f setropts.mod
```

issues a RACF **setropts** command to refresh the raclisted profiles in the facility class.

Note:

1. The PROGRAM class is not refreshed using RACLIST. Instead, issue the **ldapmodify** command with `setropts.mod` containing:

```
dn: cn=setropts,sysplex=sysplexa
changetype: modify
racfsetroptsattributes: when(program)
racfsetroptsattributes: refresh
```

2. Refreshing the CFIELD class using SDBM is not supported.

Deleting attributes

If a request is made to delete the **racfAccessControl, racfAttributes, racfConnectAttributes, racfResourceAttributes**, or **racfSetroptsAttributes** attribute and no values are provided, SDBM adds the following to the appropriate RACF command (even if the profile does not currently have that value):

- **racfAccessControl** - RESET(ALL)
- **racfAttributes** - NOADSP NOAUDITOR NOGRPACC NOOIDCARD NOOPERATIONS NOSPECIAL NOUAUDIT
- **racfConnectAttributes** - NOADSP NOAUDITOR NOGRPACC NOOPERATIONS NOSPECIAL
- **racfResourceAttributes** - NOSINGLEDSN NOTVTOC NOWARNING
- **racfSetroptsAttributes** - NOWHEN(PROGRAM)

It is required to specify the value itself on the delete operation when deleting a specific value for these attributes.

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), 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`, 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`, 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 18. Password policy

Password policy is a set of rules that controls how passwords are used and administered in the LDAP server. These password policy rules are enforced to ensure that password values are changed periodically and meet the syntactic password requirements of your organization. These rules also restrict the reuse of old passwords, ensure that users are locked out after a defined number of failed bind attempts, and automatically expire passwords after a period of time.

The LDAP password policy rules only apply to entries that have a **userPassword** value stored in a TDBM, LDBM, or CDBM backend. Entries that are outside of a configured backend suffix and have their password values stored in the LDAP server configuration file rather than in a TDBM, LDBM, or CDBM are not subject to LDAP password policy. These users include the LDAP root administrator defined in the configuration file (**adminDN** configuration option) when the password value is specified in the **adminPW** configuration option, the master server DN when the password is specified in the **masterServerPW** configuration option, and the peer server DN when the password value is specified in the **peerServerPW** configuration option.

LDAP password policy is checked during authentication and compare operations involving the **userPassword** attribute value to ensure that the password has not expired or the user's account has not been locked from authenticating to the directory. The only supported bind mechanisms for password policy checking are simple, CRAM-MD5, and DIGEST-MD5 when the authenticating user's entry and password resides in a TDBM, LDBM, or CDBM backend. Because LDAP password policy is checked during simple, CRAM-MD5, and DIGEST-MD5 authentications and compare operations involving the **userPassword** attribute value, when the term, authentication, is referenced in this section, it indicates each of these scenarios. LDAP password policy is not checked during anonymous, Kerberos (GSSAPI), or EXTERNAL binds as these authentication mechanisms do not access a password value. LDAP password policy also does not apply to TDBM, LDBM, or CDBM entries participating in native authentication or entries in the SDBM backend. The z/OS security manager handles the password policy for these users. See “Binding with SDBM using password policy” on page 329 and “Password policy with native authentication” on page 407 for more information.

During add and modify requests of password values in the TDBM, LDBM, and CDBM backends, the LDAP password policy is checked to verify that the password syntax is correct, the password is allowed to be changed now, and if the password must be changed currently for the user.

Password policy entries

The server compatibility level must be 6 or greater and the CDBM backend must be configured to use LDAP password policy. See “serverCompatLevel {3 | 4 | 5 | 6 | 7| 8}” on page 125 for more information about the **serverCompatLevel** configuration option. When the server compatibility level is 6 or greater and the CDBM backend is configured, the LDAP server automatically creates the **cn=pwdpolicy,cn=ibmpolicies** entry in the CDBM backend, if it does not exist.

The **cn=pwdpolicy,cn=ibmpolicies** entry is also known as the global password policy entry, and it controls if password policy is active in the LDAP server. By

default, the global password policy is not active (set to false), but is activated by setting the **ibm-pwdPolicy** attribute value to true. When activated, the global password policy entry is the default password policy and applies to all TDBM, LDBM, and CDBM entries that have a **userPassword** attribute value. The default values specified in Table 45 on page 367 are the default values of the global password policy entry.

If an individual or group needs to use a special password policy that is different from the global password policy, additional password policy entries are added under the **cn=ibmpolicies** suffix in the CDBM backend. In these additional password policy entries, it is only necessary to specify a password policy attribute value if it is different from the value specified in the global password policy or the default for the attribute value. Depending on the password policy attributes used in these additional password policy entries, an objectclass attribute value of **pwdPolicy** or **ibm-pwdPolicyExt** is required. Because these objectclasses are auxiliary, it is necessary to include a structural objectclass value, such as **container**, to these password policy entries. See Table 45 on page 367 for information about the password policy attribute types.

These additional password policy entries are allowed to be referenced by individual or group entries in the directory. The distinguished name of a password policy entry is referenced by a static, dynamic, and nested group by adding or modifying the single-valued **ibm-pwdGroupPolicyDN** operational attribute value in a group entry. See Chapter 23, “Static, dynamic, and nested groups,” on page 419 for more information about group entries. When referenced by a group entry, these password policy entries are referred to as group password policy entries. The distinguished name of a password policy entry is also referenced by an individual user by adding or modifying the single-valued **ibm-pwdIndividualPolicyDN** operational attribute value in a user entry. These password policy entries are referred to as individual password policy entries. Multiple users and groups can point to the same password policy entries. See “Password policy examples” on page 387 for examples on modifying user and group entries to reference password policies.

Individual and group password policy entries are only activated when the **ibm-pwdPolicy** attribute is set to true in their own entries and the **ibm-pwdGroupAndIndividualEnabled** attribute in the global password policy entry is set to true.

Note:

1. If the **ibm-pwdIndividualPolicyDN** attribute value is `cn=noPwdPolicy` in a user entry, that user is exempt from any password policy controls. A user can also be exempted from password policy controls if the **ibm-pwdGroupPolicyDN** attribute value is `cn=noPwdPolicy` in the user's groups.
2. The password policy entry must be created before it can be referenced by a user or group entry as an individual or group password policy. When a password policy entry is referenced by any user or group entry in an **ibm-pwdIndividualPolicyDN** and **ibm-pwdGroupPolicyDN** attribute value, the password policy entry cannot be renamed or deleted until all references to the password policy distinguished name (DN) are removed from all individual and group entries.
3. Password policy entries must be under the **cn=ibmpolicies** suffix in the CDBM backend. Entries located elsewhere in the directory are ignored.

See “Password policy evaluation” on page 375 for information about determining the effective password policy if a user belongs to individual and multiple password policy groups.

Activating password policy

When the global password policy entry, **cn=pwdpolicy,cn=ibmpolicies**, is initially created, each attribute value is assigned the default value specified in Table 45. If the global password policy is activated and these defaults are not updated, existing users are required to change their passwords on their next successful authentication to the LDAP server. The reason is because the default value for the **pwdMustChange** attribute in the global password policy entry is true that indicates users must change their password values. Before activating the global password policy entry, carefully consider changing the **pwdMustChange** attribute value to false in the global password policy if it is not necessary to force all existing users to change their password values.

When the global password policy is activated, user entries are allowed to have only one **userPassword** attribute value. If there are existing users that have multiple passwords before the global password policy is activated, password policy checking is completed based on the global password policy configuration. When changing password values for users that have multiple **userPassword** values, all existing password values must be replaced with just one **userPassword** value.

Password policy attributes

The password policy attributes in Table 45 are specified in individual and group password policy entries. When the global password policy entry is automatically created by the server, it uses the default values specified in this table. If a password policy attribute is not specified in an entry, it defaults to the value determined by the password policy evaluation rules. Table 45 describes the most restrictive value for each attribute for password policy evaluation. See “Password policy evaluation” on page 375 for more information.

Table 45. Password policy attributes and default values

Attribute description and example
ibm-pwdGroupAndIndividualEnabled A boolean (true or false) indicating if group and individual password policy entries are to be considered when evaluating password policy. When the global password policy entry is initially created by the server, this attribute is set to false indicating that only the global policy is used when evaluating password policy. If set to true, the global, group, and individual password policies are to be considered when evaluating password policy. This is a required attribute of the **ibm-pwdGroupAndIndividualPolicies** objectclass and is only evaluated in the global password policy entry. **Default:** false Example: `ibm-pwdGroupAndIndividualEnabled: false`

Table 45. Password policy attributes and default values (continued)

Attribute description and example
ibm-pwdPolicy A boolean (true or false) indicating if this password policy entry is active. If set to true, the password policy rules in this entry are enforced. If set to false, the password policy rules in this entry are not enforced. If this is an individual or group password policy entry, the **ibm-pwdPolicy** and the **ibm-pwdGroupAndIndividualEnabled** attributes in the global password policy, **cn=pwdpolicy,cn=ibmpolicies**, must be set to true to allow activation or evaluation of this password policy entry. This is an optional attribute of the **ibm-pwdPolicyExt** objectclass. **Default:** false **Most restrictive value for composite group evaluation:** true Example: `ibm-pwdPolicy: true`
ibm-pwdPolicyStartTime Specifies the time in Zulu format when an LDAP administrator activated this password policy entry. When the **ibm-pwdPolicy** attribute is set to true, this attribute automatically resets to the current time. This value is used during password expiration checking. This is an optional attribute of the **ibm-pwdPolicyExt** objectclass. **Most restrictive value for composite group evaluation:** The oldest defined time is used. Example: `ibm-pwdPolicyStartTime: 20091001170933.867354Z`
passwordMinAlphaChars Specifies the minimum number of alphabetic characters (A-Z and a-z) that a password value must have. It must be set to less than or equal to the value in the **pwdMinLength** attribute minus the value in the **passwordMinOtherChars** attribute. If set to 0, there is no minimum number of alphabetic characters that a password value must have. If the number of alphabetic characters cannot be checked (because of a one-way hashed password or other reasons), based on the value of the **pwdCheckSyntax** attribute, the password is accepted without checking it ('0' or '1') or is refused ('2'). This is an optional attribute of the **ibm-pwdPolicyExt** objectclass. **Default:** 0 **Most restrictive value for composite group evaluation:** See “Evaluation of a user's individual and composite group password policy” on page 375 for more information. Example: `passwordMinAlphaChars: 3`

Table 45. Password policy attributes and default values (continued)

Attribute description and example
passwordMinOtherChars Specifies the minimum number of numeric and special characters (other than A-Z and a-z) that a password value must have. It must be set to less than or equal to the value in **pwdMinLength** attribute minus the value in **passwordMinAlphaChars** attribute. If set to 0, there is no minimum number of other characters that a password value must have. If the number of numeric and special characters cannot be checked (because of a one-way hashed encrypted password or other reasons), based on the value of the **pwdCheckSyntax** attribute, either the password is accepted without checking it ('0' or '1') or is refused ('2'). This is an optional attribute of the **ibm-pwdPolicyExt** objectclass. **Default:** 0 **Most restrictive value for composite group evaluation:** See "Evaluation of a user's individual and composite group password policy" on page 375 for more information. Example: `passwordMinOtherChars: 2`
passwordMaxRepeatedChars Specifies the maximum number of times a given character is used in a password value. If set to 0, there is no limit on the number of repeated characters that a password value must have. If the maximum number of repeated characters cannot be checked (because of a one-way hashed encrypted password or other reasons), based on the value of the **pwdCheckSyntax** attribute, either the password is accepted without checking it ('0' or '1') or is refused ('2'). This is an optional attribute of the **ibm-pwdPolicyExt** objectclass. **Default:** 0 **Most restrictive value for composite group evaluation:** See "Evaluation of a user's individual and composite group password policy" on page 375 for more information. Example: `passwordMaxRepeatedChars: 2`
passwordMaxConsecutiveRepeatedChars Specifies the maximum number of successive repetitions of a given character in a password value. If set to 0, the number of consecutive repeated characters is not checked in the password value. If the maximum number of consecutive repeated characters cannot be checked (because of a one-way hashed encrypted password or other reasons), based on the value of the **pwdCheckSyntax** attribute, either the password is accepted without checking it ('0' or '1') or is refused ('2'). For example, if set to 1, a given character cannot be followed by another character of the same type. Therefore, 'aba' is valid and 'aab' is not valid. For example, if set to 2, the maximum number of times a given character can occur consecutively is 2. Therefore, 'aaba' is valid and 'aaab' is not valid. This is an optional attribute of the **ibm-pwdPolicyExt** objectclass. **Default:** 0 **Most restrictive value for composite group evaluation:** See "Evaluation of a user's individual and composite group password policy" on page 375 for more information. Example: `passwordMaxConsecutiveRepeatedChars: 2`

Table 45. Password policy attributes and default values (continued)

Attribute description and example
passwordMinDiffChars Specifies the minimum number of characters in the new password that must be different from the characters in the old password value. If set to 0, the number of different characters are not checked in the new password value. If the password characters cannot be checked (because of a one-way hashed encrypted password or other reasons), based on the value of the **pwdCheckSyntax** attribute, either the password is accepted without checking it ('0' or '1') or is refused ('2'). This is an optional attribute of the **ibm-pwdPolicyExt** objectclass. **Default:** 0 **Most restrictive value for composite group evaluation:** See "Evaluation of a user's individual and composite group password policy" on page 375 for more information. Example: `passwordMinDiffChars: 0`
pwdAllowUserChange A boolean (true or false) indicating if users are allowed to change their own passwords. If set to false, users are not allowed to change their own password. If set to true, users are allowed to change their own password if they have the authority. If the **pwdMustChange** attribute is set to true, this attribute must also be set to true to allow users to change their passwords. This is an optional attribute of the **pwdPolicy** objectclass. **Default:** true **Most restrictive value for composite group evaluation:** false Example: `pwdAllowUserChange: false`
pwdAttribute Specifies the name of the attribute this password policy applies to. This attribute is only allowed to be set to **userPassword**. If this attribute is attempted to be set to another value, an error is returned on the add or modify request of the password policy entry. This is a required attribute of the **pwdPolicy** objectclass. **Default: userPassword** Example: `pwdAttribute: userPassword`

Table 45. Password policy attributes and default values (continued)

Attribute description and example
pwdCheckSyntax Specifies if the user's password value is checked for password policy syntax when adding or modifying a **userPassword** attribute value. The password policy attributes that affect password policy syntax are **passwordMinAlphaChars, passwordMinOtherChars, passwordMaxRepeatedChars, passwordMaxConsecutiveRepeatedChars, passwordMinDiffChars**, and **pwdMinLength**. If set to 0, syntax checking of the password value is not enforced. If set to 1, the password syntax is checked and if it cannot be checked (because of a one-way hashed encrypted password or other reasons) it is accepted. If set to 2, the password syntax is checked and if it cannot be checked (because of a one-way hashed password or other reasons), an error is returned on the add or modify request of the user's password value. For example, if the current **userpassword** stored in the directory is one-way hashed, the **passwordMinDiffChars** comparison of the new and current password cannot be performed. This is an optional attribute of the **pwdPolicy** objectclass. **Note:** Password policy syntax checking is not enforced when an LDAP root or password administrator is adding or modifying another user's **userPassword** attribute value. If the LDAP root administrator defined in the configuration file (**adminDN** configuration option) is updating its own **userPassword** attribute value, password policy syntax checking is enforced. **Default:** 0 **Most restrictive value for composite group evaluation:** The largest value is used. Example: `pwdCheckSyntax: 2`
pwdExpireWarning Specifies the number of seconds before a password is due to expire that expiration warning messages are returned during authentication in the **PasswordPolicy** response control. If set to 0, password policy expiration warnings are not returned during authentication in the **PasswordPolicy** response control. This is an optional attribute of the **pwdPolicy** objectclass. **Default:** 0 **Most restrictive value for composite group evaluation:** The largest value is used. Example: `pwdExpireWarning: 600`
pwdFailureCountInterval Specifies the number of seconds when password failures are removed from the failure counter although no successful authentication occurred. If set to 0, the failure counter is only reset by a successful authentication. This is an optional attribute of the **pwdPolicy** objectclass. **Default:** 0 **Most restrictive value for composite group evaluation:** The largest value is used. Example: `pwdFailureCountInterval: 600`

Table 45. Password policy attributes and default values (continued)

Attribute description and example
pwdGraceLoginLimit Specifies the number of times an expired password is used for authentication. If set to 0 and the password expired, authentication fails. When the number of grace logins or authentications is exceeded by the user, the password is automatically expired and authentication fails. There is no time limit that is applied to this attribute. This is an optional attribute of the **pwdPolicy** objectclass. **Default:** 0 **Most restrictive value for composite group evaluation:** The smallest value is used. Example: `pwdGraceLoginLimit: 3`
pwdInHistory Specifies the maximum number of used password values that are stored in the **pwdHistory** operational attribute of user entries. If set to 0, used password values are not stored in the **pwdHistory** attribute of user entries, therefore, they might be reused. The maximum value for this attribute is 30. The passwords that are stored in the **pwdHistory** operational attribute values are used to check new password values that are modified by the user. This is an optional attribute of the **pwdPolicy** objectclass. **Default:** 0 **Most restrictive value for composite group evaluation:** The largest value is used. Example: `pwdInHistory: 15`
pwdLockout A boolean (true or false) used to indicate if a password might be used for authentication when the maximum number of consecutive failed authentication attempts specified in the **pwdMaxFailure** attribute is exceeded. If set to true, the user's account is eligible to be locked for the number of seconds specified in the **pwdLockoutDuration** attribute value. If set to false, the user's account is not eligible to be locked. This is an optional attribute of the **pwdPolicy** objectclass. **Default:** false **Most restrictive value for composite group evaluation:** true Example: `pwdLockout: true`

Table 45. Password policy attributes and default values (continued)

Attribute description and example
pwdLockoutDuration Specifies the number of seconds a password cannot be used for authentication when the maximum number of consecutive failed authentication attempts in the **pwdMaxFailure** attribute is exceeded and the **pwdLockout** attribute is set to true. If set to 0 and the maximum number of consecutive failed authentication attempts is exceeded, the password cannot be used for authentication until it is reset by an LDAP root or password administrator. This is an optional attribute of the **pwdPolicy** objectclass. **Default:** 0 **Most restrictive value for composite group evaluation:** The largest value is used. Example: `pwdLockoutDuration: 600`
pwdMaxAge Specifies the maximum age, in seconds, when a modified password expires. If set to 0, the password value does not expire for the user. This value must be greater than or equal to the **pwdMinAge** attribute value. This is an optional attribute of the **pwdPolicy** objectclass. **Default:** 0 **Most restrictive value for composite group evaluation:** The smallest value is used. Example: `pwdMaxAge: 6000`
pwdMaxFailure Specifies the maximum number of consecutive failed authentication attempts allowed. If set to 0, there is no maximum number of consecutive failed authentication attempts and the **pwdLockout** attribute value is ignored. If greater than 0, the **pwdLockout** attribute is set to true, and the maximum number of failed authentication attempts is reached, authentication is not allowed until the lockout time specified in the **pwdLockoutDuration** attribute value is exceeded. This is an optional attribute of the **pwdPolicy** objectclass. **Default:** 0 **Most restrictive value for composite group evaluation:** The smallest value is used. Example: `pwdMaxFailure: 5`
pwdMinAge Specifies the number of seconds that must elapse between modifications to the password. If set to 0, the password value is modified without waiting for time to elapse. This value must be less than or equal to the **pwdMaxAge** value. This is an optional attribute of the **pwdPolicy** objectclass. **Default:** 0 **Most restrictive value for composite group evaluation:** The largest value is used. Example: `pwdMinAge: 600`

Table 45. Password policy attributes and default values (continued)

Attribute description and example
pwdMinLength Specifies the minimum length of the password value for the user. If set to 0, there is no minimum password value length checking enforced. If the password length cannot be checked (because of a one-way hashed password or other reasons), based on the value of the **pwdCheckSyntax** attribute, either the password is accepted without checking it ('0' or '1') or is refused ('2'). This is an optional attribute of the **pwdPolicy** objectclass. **Default:** 0 **Most restrictive value for composite group evaluation:** See "Evaluation of a user's individual and composite group password policy" on page 375 for more information. Example: `pwdMinLength: 8`
pwdMustChange A boolean (true or false) indicating users must change their passwords after their first successful authentication to the server after a password is set or reset by an LDAP root or password administrator. If set to true, users are required to change their password after their first successful authentication. If set to false, users are not required to change their password upon successful authentication when the password is set or reset by an LDAP root or password administrator. If this attribute is set to true, the **pwdAllowUserChange** attribute must also be set to true to allow users to change their password values. This is an optional attribute of the **pwdPolicy** objectclass. **Default:** true **Most restrictive value for composite group evaluation:** true Example: `pwdMustChange: true`
pwdSafeModify A boolean (true or false) indicating the existing or current password value must be sent when changing a password value. If set to true, the existing password value must be sent on the modify request. If set to false, the existing password value does not need to be sent on the modify request. The **ldapchangepwd** utility can be used to change a user's password value. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information about the **ldapchangepwd** utility. This is an optional attribute of the **pwdPolicy** objectclass. **Default:** false **Most restrictive value for composite group evaluation:** true Example: `pwdSafeModify: true`

Note: When the **ibm-pwdPolicy** attribute is changed from false to true in the password policy entry, the **ibm-pwdPolicyStartTime** attribute is automatically set by the server to the current time. This password policy start time is used for performing password expiration and age checking of a user's password value if either of these password policy features are enabled for the user. When a password value is modified, the **pwdChangedTime** attribute value in the user's entry is updated with the time of the password modification. The password policy start

time and the last password modification time are used to determine if the user's password expired. However, if the **ibm-pwdPolicy** is changed from false to true later, user passwords that are scheduled to expire might no longer expire at the original time because the password policy start time is set to a more recent time.

Password policy evaluation

Because users can belong to an individual password policy, belong to multiple groups that have varying password policies, and the global password policy, there are a set of rules that are followed for determining the user's effective password policy when conflicting password policies are in effect.

Individual and group password policies are not evaluated when determining a user's effective password policy until the **ibm-pwdPolicy** and the **ibm-pwdGroupAndIndividualEnabled** attribute values are set to true in the global password policy. In addition, each individual or group password policy is not activated or evaluated for determining a user's effective password policy until the **ibm-pwdPolicy** attribute value is set to true in those entries.

When the password policies are activated, the different levels of password policies (individual, group, and global) are accessed when evaluating a user's effective password policy. The password policies are evaluated in this order, if they exist:

- individual policy entry
- composite group policy entry
- global password policy entry
- default password policy attribute value

Typically, when a password policy attribute is found in any of the password policy entries, the evaluation of that attribute stops and that attribute value is used as part of the user's effective password policy. See Table 45 on page 367 for the default password policy attribute values.

Evaluation of a user's individual and composite group password policy

A user entry may belong to an individual password policy entry. Because the user also might belong to more than one group, multiple group password policy entries may be evaluated to determine a user's composite group policy. These are the rules for determining a user's effective password policy:

1. If the **ibm-pwdPolicy** attribute value is false or is not specified in the individual or group password policy entry, no attributes defined in the entry are used to determine the effective password policy.
2. If the **ibm-pwdIndividualPolicyDN** attribute value is `cn=noPwdPolicy` in a user entry, that user is exempt from any password policy controls. A user can also be exempted from any password policy controls if all the user's groups that have the **ibm-pwdGroupPolicyDN** attribute have a value of `cn=noPwdPolicy` and the user entry does not have an **ibm-pwdIndividualPolicyDN**. If any of the user's groups has the **ibm-pwdGroupPolicyDN** attribute referring to an active policy, then references to `cn=noPwdPolicy` in the user's other groups are ignored in the composite group policy evaluation.
3. The **ibm-pwdPolicyStartTime** attribute is set to the current system time when **ibm-pwdPolicy** is set to true in the password policy entry. It is set to a new time whenever its value is changed from false to true. In determining the **ibm-pwdPolicyStartTime** value for a composite group, the oldest defined value is used.

4. The password policy attributes **passwordMinAlphaChars, pwdMinLength**, and **passwordMinOtherChars** are interdependent. For example, the value of **passwordMinAlphaChars** must be less than or equal to the value in **pwdMinLength** and **pwdMinDiffChars** minus the value in **passwordMinOtherChars**. Because of this interdependency among attribute values, if one attribute is selected from a policy, then the other two attributes are also selected from the same policy.

   If an individual password policy is not enabled and there are multiple group password policies enabled, the order of selection is **pwdMinLength, passwordMinOtherChars, passwordMinAlphaChars,** and **pwdMinDiffChars**. This means that the selection is based on picking the most restrictive (the largest value) for **pwdMinLength**. If multiple group policies have the same largest value for the **pwdMinLength** attribute, then the one with the most restrictive (the largest value) for **passwordMinOtherChars** is selected. If multiple group policies also have the same largest **passwordMinOtherChars** value, then the group policy with the most restrictive (the largest value) **pwdMinDiffChars** is used. After an attribute is selected, the other three attributes are selected automatically from that same policy.

5. The password policy attributes **pwdMinAge, pwdMaxAge**, and **pwdExpirationWarning** are interdependent. For example, the value of **pwdMinAge** must be less than or equal to the value in **pwdMaxAge** and the value of **pwdExpireWarning** must be less than or equal to the value in **pwdMaxAge**. Because of this interdependency among attribute values, if one attribute is selected from a policy, then the other two attributes are also selected from the same policy.

   If an individual password policy is not enabled and there are multiple group policies enabled, the order of selection is **pwdMaxAge, pwdMinAge,** and **pwdExpireWarning**. This means that the selection is based on picking the most restrictive value (the smallest value) for **pwdMaxAge**. In a situation where multiple group policies have the same value for the **pwdMaxAge** attribute, then the one with the most restrictive value (the largest value) for **pwdMinAge** is selected. In a situation where multiple group policies have the same value for the **pwdMaxAge** and **pwdMinAge** attributes, then the one with the most restrictive value (the largest value) for **pwdExpireWarning** is selected. After an attribute is selected, the other two attributes are selected automatically from that same policy.

6. The **passwordMaxConsecutiveRepeatedChars** attribute is used to restrict the maximum successive repetitions of a given character in the password. **passwordMaxRepeatedChars** and **passwordMaxConsecutiveRepeatedChars** attributes are allowed to be enabled or disabled, independent of each other. However, if both of these attributes are enabled, these rules are applicable:
   - The value of the **passwordMaxRepeatedChars** attribute must be greater than or equal to the value of the **passwordMaxConsecutiveRepeatedChars** attribute.
   - When multiple group password policies are enabled, the **passwordMaxConsecutiveRepeatedChars** value is obtained from the same policy where the **passwordMaxRepeatedChars** attribute value is obtained from. If **passwordMaxRepeatedChars** is disabled in all policies, then the most restrictive value of **passwordMaxConsecutiveRepeatedChars** is used.

7. The password policy attributes **pwdMustChange** and **pwdAllowUserChange** are interdependent. For example, setting the value of **pwdMustChange** to true and **pwdAllowUserChange** to false is not allowed, as this says that a user must change their password after it has been reset by an administrator, but does not give the user the ability to change their password. Because of this, when

adding or modifying a policy, this combination is checked for and is not allowed. Therefore, considering this interdependency between the attributes, if one attribute is selected from a policy, then the other attribute is also selected from the same policy.

The default for the **pwdMustChange** and **pwdAllowUserChange** attributes is true. The most restrictive value for the **pwdMustChange** attribute is true. The most restrictive value for the **pwdAllowUserChange** attribute is false. When choosing which group policy supplies these two attributes, the one with the most restrictive value for **pwdAllowUserChange** is chosen. If all the values for **pwdAllowUserChange** are true, then the group with the most restrictive value for **pwdMustChange** is chosen.

8. Attributes in all active group password policy entries are combined to form a union of attributes with the most restrictive attribute values taking precedence. See "Password policy attributes" on page 367 and Table 46 for information about how the most restrictive attribute values are determined.

To better understand how a composite group policy is determined, consider some examples given in Table 46. The examples in Table 46 only display values for the attributes specified in the various groups, not all the default values for the attributes that are not specified.

Table 46. Composite group password policy examples

Group X password policy	Group Y password policy	Group Z password policy	Composite group password policy
pwdMaxAge = 86400 pwdSafeMode = true pwdMaxFailure = 5 ibm-pwdPolicy = true ibm-pwdPolicyStartTime = 20090908153021.242838Z	pwdMaxAge = 43200 pwdSafeMode = false ibm-pwdPolicy = true ibm-pwdPolicyStartTime = 20090808153021.242838Z	pwdCheckSytax = 1 ibm-pwdPolicy = true ibm-pwdPolicyStartTime = 20091008153021.242838Z	pwdMaxAge = 43200 pwdSafeMod = true pwdCheckSytax = 1 pwdMaxFailure = 5 ibm-pwdPolicy = true ibm-pwdPolicyStartTime = 20090808153021.242838Z
pwdMaxAge = 86400 ibm-pwdPolicy = true	pwdMaxAge = 43200 pwdSafeMode = true	pwdMaxAge = 0 ibm-pwdPolicy = true	pwdMaxAge = 86400 pwdSafeMode = false ibm-pwdPolicy = true **Note:** Group Y's password policy is not used in calculating the composite group policy because it does not have an **ibm-pwdPolicy** attribute and, therefore, it defaults to false.
pwdMinLength = 10 passwordMinOtherChars = 4 passwordMinAlphaChars = 6 ibm-pwdPolicy = true	pwdMinLength = 12 ibm-pwdPolicy = true		pwdMinLength = 12 ibm-pwdPolicy = true
pwdMinLength = 10 passwordMinOtherChars = 4 passwordMinAlphaChars = 6 ibm-pwdPolicy = true		pwdMinLength = 10 passwordMinOtherChars = 5 passwordMinAlphaChars = 3 ibm-pwdPolicy = true	pwdMinLength = 10 passwordMinOtherChars = 5 passwordMinAlphaChars = 3 ibm-pwdPolicy = true
passwordMaxConsecutiveRepeatedChars = 0 passwordMaxRepeatedChars = 5 ibm-pwdPolicy = true	passwordMaxConsecutiveRepeatedChars = 2 ibm-pwdPolicy = true	passwordMaxRepeatedChars = 3 ibm-pwdPolicy = true	passwordMaxRepeatedChars = 3 passwordMaxConsecutiveRepeatedChars = 0 ibm-pwdPolicy = true

Table 46. Composite group password policy examples (continued)

Group X password policy	Group Y password policy	Group Z password policy	Composite group password policy
passwordMaxConsecutiveRepeatedChars = 4 passwordMaxRepeatedChars = 0 ibm-pwdPolicy = true	passwordMaxConsecutiveRepeatedChars = 1 passwordMaxRepeatedChars = 0 ibm-pwdPolicy = true		passwordMaxConsecutiveRepeatedChars = 1 passwordMaxRepeatedChars = 0 ibm-pwdPolicy = true
passwordMaxConsecutiveRepeatedChars = 4 passwordMaxRepeatedChars = 2 ibm-pwdPolicy = true	passwordMaxConsecutiveRepeatedChars = 2 passwordMaxRepeatedChars = 3 ibm-pwdPolicy = true		passwordMaxConsecutiveRepeatedChars = 4 passwordMaxRepeatedChars = 2 ibm-pwdPolicy = true
pwdMaxAge = 200000 pwdExpireWarning = 100000 ibm-pwdPolicy = true		pwdMaxAge = 200000 pwdExpireWarning = 150000 ibm-pwdPolicy = true	pwdMaxAge = 200000 pwdExpireWarning = 150000 ibm-pwdPolicy = true
pwdMaxAge = 200000 pwdMinAge = 100000 ibm-pwdPolicy = true	pwdMaxAge = 200000 pwdExpireWarning = 100000 ibm-pwdPolicy = true	pwdMaxAge = 200000 pwdExpireWarning = 150000 ibm-pwdPolicy = true	pwdMaxAge = 200000 pwdMinAge = 100000 pwdExpireWarning = 0 ibm-pwdPolicy = true
pwdMaxAge = 100000 pwdMinAge = 10000 pwdExpireWarning = 50000 ibm-pwdPolicy = true	pwdMaxAge = 100000 pwdMinAge = 50000 pwdExpireWarning = 70000 ibm-pwdPolicy = true	pwdMaxAge = 100000 pwdMinAge = 50000 pwdExpireWarning = 80000 ibm-pwdPolicy = true	pwdMaxAge = 100000 pwdMinAge = 50000 pwdExpireWarning = 80000 ibm-pwdPolicy = true

Effective password policy examples

The **Effective password policy** extended operation is useful for an LDAP administrator to query a user's effective password policy when there are multiple password policies active. This extended operation is supported in the **ldapexop** utility. See “ldapexop utility” on page 255 for more information.

Table 47 displays examples of how a user's effective password policy is determined when individual, group, and global password policies are active.

Table 47. Effective password policy examples

Individual password policy	Composite group password policy	Global password policy	Effective password policy
ibm-pwdpolicy=true ibm-pwdpolicystarttime= 20091102173246.353496Z pwdminlength=8 pwdgraceloginlimit=3 pwdinhistory=4 pwdattribute=userpassword	ibm-pwdpolicy=true ibm-pwdpolicystarttime= 20091102173256.250491Z pwdexpirewarning=2592000 pwdminlength=10 pwdinhistory=5 pwdchecksyntax=2 passwordminalphachars=5 passwordminotherchars=2 pwdattribute=userpassword pwdmaxage=5184000	ibm-pwdpolicy=true ibm-pwdpolicystarttime= 20091102163354.065965Z pwdgraceloginlimit=0 pwdsafemodify=FALSE pwdlockoutduration=0 pwdmaxfailure=5 pwdfailurecountinterval=0 pwdmaxage=7776000 pwdexpirewarning=5184000 pwdminlength=5 pwdlockout=true pwdallowuserchange=TRUE pwdmustchange=TRUE ibm-pwdgroupandindividualenabled=true passwordmaxconsecutiverepeatedchars=0 passwordmaxrepeatedchars=0 passwordminalphachars=0 passwordminotherchars=0 passwordmindiffchars=0 pwdminage=0 pwdinhistory=3 pwdchecksyntax=0	ibm-pwdpolicy=TRUE ibm-pwdpolicystarttime= 20091102173246.353496Z passwordmaxconsecutiverepeatedchars=0 passwordmaxrepeatedchars=0 passwordminalphachars=0 passwordmindiffchars=0 passwordminotherchars=0 pwdallowuserchange=TRUE pwdattribute=userpassword pwdchecksyntax=2 pwdexpirewarning=2592000 pwdfailurecountinterval=0 pwdgraceloginlimit=3 pwdinhistory=4 pwdlockout=TRUE pwdlockoutduration=0 pwdmaxage=5184000 pwdmaxfailure=5 pwdminage=0 pwdminlength=8 pwdmustchange=TRUE pwdsafemodify=FALSE

Table 47. Effective password policy examples (continued)

Individual password policy	Composite group password policy	Global password policy	Effective password policy
ibm-pwdpolicy=true ibm-pwdpolicystarttime= 20091103132230.734257Z pwdgraceloginlimit=8 pwdmaxage=2592000 pwdminage=86400	ibm-pwdpolicy=true ibm-pwdpolicystarttime= 20091103131723.827250Z pwdminlength=10 pwdgraceloginlimit=10 pwdmaxage=5184000 pwdminage=86400 pwdexpirewarning=2592000 passwordmindiffchars=2 passwordminotherchars=0 pwdinhistory=5	ibm-pwdpolicy=true ibm-pwdpolicystarttime= 20091102163354.065965Z pwdgraceloginlimit=10 pwdsafemodify=FALSE pwdlockoutduration=0 pwdmaxfailure=5 pwdfailurecountinterval=0 pwdmaxage=7776000 pwdexpirewarning=5184000 pwdminlength=5 pwdlockout=true pwdallowuserchange=TRUE pwdmustchange=TRUE ibm-pwdgroupandindividualenabled=true passwordmaxconsecutiverepeatedchars=0 passwordmaxrepeatedchars=0 passwordminalphachars=0 passwordminotherchars=1 passwordmindiffchars=3 pwdminage=2592000 pwdinhistory=0 pwdchecksyntax=0	ibm-pwdpolicy=TRUE ibm-pwdpolicystarttime= 20091103132230.734257Z ibm-pwdgroupandindividualenabled=TRUE passwordmaxconsecutiverepeatedchars=0 passwordmaxrepeatedchars=0 passwordminalphachars=0 passwordmindiffchars=2 passwordminotherchars=0 pwdallowuserchange=TRUE pwdattribute=userpassword pwdchecksyntax=0 pwdexpirewarning=0 pwdfailurecountinterval=0 pwdgraceloginlimit=8 pwdinhistory=5 pwdlockout=TRUE pwdlockoutduration=0 pwdmaxage=2592000 pwdmaxfailure=5 pwdminage=86400 pwdminlength=10 pwdmustchange=TRUE pwdsafemodify=FALSE

In the first row of Table 47 on page 378, the **pwdMinLength, pwdGraceloginlimit**, and **pwdInHistory** values are selected from the individual policy as that policy is evaluated first. When the **pwdMinLength** value is selected from the individual policy, the **passwordMinAlphaChars** and **passwordMinOtherChars** values default to 0 because these attributes are interdependent and must come from the same policy. The **pwdExpireWarning, pwdCheckSyntax, passwordMinAlphaChars, passwordMinOtherChars**, and **pwdMaxAge** are selected from the composite group policy because that is evaluated next. The remaining password attribute values are selected from the global password policy.

In the second row of Table 47 on page 378, the **pwdGraceLoginLimit, pwdMaxAge**, and **pwdMinAge** are selected from the individual policy because those policies are evaluated first. When the **pwdMaxAge** and **pwdMinAge** are selected from the individual policy, the **pwdExpireWarning** defaults to 0 because these attributes are interdependent and must come from the same policy. The **pwdMinLength, passwordMinDiffChars, passwordMinOtherChars**, and **pwdInHistory** are selected from the composite group because that is evaluated next. The remaining password attribute values are selected from the global password policy.

Password policy operational attributes

For user entries that are subject to LDAP password policy, there are several operational attributes that contain password policy state information. The password operational attributes in Table 48 on page 380 are in the critical access class and by default, only an LDAP administrator with the appropriate authority can query and read them. See Chapter 9, “Administrative group and roles,” on page 159 for more information. If other users require access to these attributes, the default ACLs may be changed to allow read access to these attributes. See Chapter 24, “Using access control,” on page 433 for more information.

Table 48. Password policy operational attributes in user entries

Attribute and description
pwdChangedTime Specifies the Coordinated Universal Time in Zulu format when the **userPassword** value was last changed or the **ibm-pwdPolicyStartTime** attribute value of the effective password policy entry's start time whatever time is later. The **pwdChangedTime** attribute is only updated when the user's effective password policy has either the **pwdMinAge** attribute or the **pwdMaxAge** attribute set to a value other than 0. Example: `pwdChangedTime: 20091021182253.188983Z`
pwdAccountLockedTime Specifies the Coordinated Universal Time in Zulu format when the user's account was locked. If the user's account is not locked, this attribute is not present in the user's entry. If the user's password is reset by an LDAP root or password administrator, this attribute is automatically removed from the entry. Example: `pwdAccountLockedTime: 20091021183747.488417Z`
pwdExpirationWarned Specifies the Coordinated Universal Time in Zulu format of the first password expiration warning for this user. Example: `pwdExpirationWarned: 20091021181746.852469Z`
pwdFailureTime A multi-valued attribute specifying the Coordinated Universal Time in Zulu format of the previous consecutive authentication failures for this user. The number of consecutive authentication failures by this user is limited by the **pwdMaxFailure** attribute value in the effective password policy entry. On a successful authentication, all **pwdFailureTime** attribute values are removed from the user's entry. Example: `pwdFailureTime: 20091021181836.913647Z`

pwdGraceUseTime
 	
A multi-valued attribute specifying the Coordinated Universal Time in Zulu format of the previous grace logins for this user. The number of grace logins allowed by this user with an expired password is limited by the **pwdGraceLoginLimit** attribute value in the effective password policy entry. If grace logins are not allowed by the effective password policy, this attribute is not present in the user's entry. If the user's password is changed before the grace logins limit is exceeded, all **pwdGraceUseTime** attribute values are removed from the user's entry.
 	
Example:
`pwdGraceUseTime: 20091013183626.310768Z`
`pwdGraceUseTime: 20091021155707.839414Z`

pwdHistory
 	
A multi-valued attribute containing the history of previously used passwords for this user entry. The number of previous password values stored for this user is limited by the **pwdInHistory** attribute value in the effective password policy entry. When the current **userPassword** attribute value is changed for this user, the previous password values in the history are compared to ensure that the user does not reuse an old password value.
 	
The format for this attribute is:
`pwdHistory: time#syntaxOID#length#data`
 	
Where,
 	
`time` is the Coordinated Universal Time in Zulu format when this password value was added to the password history.
 	
`syntaxOID` is the numeric OID that defines the syntax used to originally store the password value.
 	
`length` is the number of octets in the old password data.
 	
`data` is the octet representing the password in a tagged base64-encoded printable format. This portion is in the same encryption or hashing format used for the original **userPassword** attribute value. If the data after the encryption tag is not printable in the original **userPassword** attribute value, it is base64 encoded before it is stored in the **pwdHistory** attribute value.
 	
Example:
`pwdHistory: 20101124182146.400909Z#1.3.6.1.4.1.1466.115.121.1.40#33#{AES:KEY}hlzJB229co0VN728TpeU0w==`
`pwdHistory: 20101124182507.050181Z#1.3.6.1.4.1.1466.115.121.1.40#62#{SSHA}eyuSkyuA1ZmIMtZWQ7WcBplgTGqiZs1X+sBV+CFOz/tynSBu3Ay=`
`pwdHistory: 20101124181038.800202Z#1.3.6.1.4.1.1466.115.121.1.40#18#{none}c2VjcmV0MQ==`

Table 48. Password policy operational attributes in user entries (continued)

Attribute and description
pwdReset A boolean (true or false) indicating if the user's password is changed or set by another user. When set to true, the password value must be changed by the user after successful authentication before the user is allowed to perform any other operations. If the **userPassword** value in this entry is changed by the user, this attribute is removed from the user's entry. Example: `pwdReset: true`
ibm-pwdAccountLocked By default, an LDAP root or directory data administrator can query these password policy operational attributes for user entries in a particular state. When a user's account is locked, the user is unable to successfully authenticate to the server. This attribute is not present in the user's entry if the account has never been administratively locked. Example: `ibm-pwdAccountLocked: true`

By default, an LDAP root or directory data administrator can query these password policy operational attributes for user entries in a particular state. Because these attributes are operational, each attribute name or the special '**+**' attribute must be specified on the search request so that they are returned on the search response. If the authenticated user has access to operational attributes, the '**+**' attribute returns all operational attributes other than the **ibm-allMembers, ibm-allGroups, ibm-entryCheckSum, ibm-entryCheckSumOp**, and **hasSubordinates** attributes.

This example uses the **ldapsearch** utility to retrieve operational attributes.

```
ldapsearch -D adminDn -w adminPw -s sub -b "c=us"  "objectclass=*" +
```

The **pwdChangedTime** attribute may be used in a search filter to retrieve a list of candidate users whose passwords may be about to expire. This example assumes that the effective password policy expiration policy is 90 days and searches for all entries of passwords expiring on August 10, 2010. Therefore, this example uses the **ldapsearch** utility to search for all entries when the password was last changed on May 2, 2010:

```
ldapsearch -D adminDn -w adminPw -s sub –b "c=us" "(!(pwdChangedTime>=201005020000000Z))" dn
```

The **pwdAccountLockedTime** attribute is used in a search filter to retrieve a list of candidate users that may be locked. Users are not always locked when this attribute is present in their entries because the effective password policy lockout duration might already be exceeded. This example uses the **ldapsearch** utility to search for all entries that have an **pwdAccountLockedTime** attribute value:

```
ldapsearch -D adminDn -w adminPw –s sub –b "c=us" "(pwdaccountlockedtime=*)" dn
```

The **pwdReset** attribute may be used in a search filter to retrieve a list of candidate users whose password must be changed because the password was reset or changed by another user. If the effective password policy does not enforce password reset, then this search does not retrieve all users that need to change or reset their passwords. This example uses the **ldapsearch** utility to search for all entries that have a **pwdReset** attribute value.

```
ldapsearch -D adminDn -w adminPw -s sub –b "c=us" "(pwdreset=true)" dn
```

PasswordPolicy control

The **PasswordPolicy** server control is specified on a client request to solicit additional warning and error information related to password policy enforcement, which the server returns in the **PasswordPolicy** response control. For example, during authentication, the **PasswordPolicy** response control can notify the user that the password must be changed, is about to expire, or there are only a few grace logins available before the user's password expires. For add requests, the **PasswordPolicy** response control provides additional error information about the password value syntax. For modify requests, the **PasswordPolicy** response control provides additional error information about the password value syntax, the new password value exists in the password history, or the old password value must be specified.

By default, the z/OS LDAP client utilities send the **PasswordPolicy** server control on these requests to obtain additional warning and error information about password policy enforcement. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information about the z/OS LDAP client utilities.

Table 49 and Table 50 contain summaries of the warnings and errors that are returned by the **PasswordPolicy** response control. See "PasswordPolicy" on page 684 for more information.

Table 49. **PasswordPolicy** *response control warnings*

Warnings and description
`timeBeforeExpiration` Indicates the number of seconds before the user's password expires.
`graceLoginsRemaining` Indicates the number of times a user is allowed to authenticate with an expired password.

Table 50. **PasswordPolicy** *response control errors*

Errors and description
`passwordExpired` Indicates that the user's password has expired and must be reset by an LDAP root administrator or an administrator with the appropriate authority. See Chapter 9, "Administrative group and roles," on page 159 for more information about administrative role authority.
`accountLocked` Indicates that the user's account is locked by an LDAP administrator with the appropriate authority, a user with sufficient authority, or the account is locked because of exceeding the maximum number of failed authentications. See Chapter 9, "Administrative group and roles," on page 159 for more information about administrative role authority.
`changeAfterReset` Indicates the user's password must be changed before the user is allowed to perform any operation other than authentication and modify requests.
`passwordModNotAllowed` Indicates that the password cannot be changed by the user even if the ACLs allow it to be changed.
`mustSupplyOldPassword` Indicates that the old password value must be supplied when changing the user's password value.

Table 50. **PasswordPolicy** *response control errors (continued)*

Errors and description
`insufficientPasswordQuality` Indicates that the new password value did not pass the quality standards specified in the effective password policy.
`passwordTooShort` Indicates that the new password value is too short.
`passwordTooYoung` Indicates that the password value is not allowed to be changed because it has been changed too recently.
`passwordInHistory` Indicates that the new password value exists in the password history and cannot be reused again.

Note: A subset of these **PasswordPolicy** response control warnings and errors are returned if the user entry is participating in native authentication or is an SDBM user. See "Binding with SDBM using password policy" on page 329 and "Password policy with native authentication" on page 407 for more information.

Replicating password policy operational attributes

If password policy operational attributes in a user's entry are updated during an add, bind, compare, or modify request, these updates are replicated to the configured replica or consumer server. These password policy operational attributes are replicated in an advanced or basic replication environment because:

- The **pwdChangedTime** attribute is replicated to all replica or consumer servers to enable expiration of the user's password on all servers.
- The **pwdReset** attribute is replicated to all replica or consumer servers to deny access to operations other than bind and modifying the user's password value.
- The **pwdHistory** attribute is replicated to all replica or consumer servers to keep the user's password history synchronized across all servers.
- The **pwdAccountLocked, pwdExpirationWarned, pwdFailureTime**, and **pwdGraceUseTime** attributes are replicated to all replica or consumer servers to enable the password policy to be managed as a whole across the entire replication environment. This allows the number of login failures, the number of grace logins, and the locking of the user's account to be managed as a whole across the entire replication environment.
- The **ibm-pwdAccountLocked** attribute is replicated to all replica or consumer servers to lock the user's entry on all servers in the replication environment.

Because these operational attributes are replicated to all consumer or replica servers, use peer-to-peer or read/write replica servers in a replication environment. This allows all servers to keep the password policy operational attribute values for user entries synchronized on all servers in the replication environment. For example, consider a master-replica replication environment with a password policy that allows users three failed logins before the user's account is locked. If users are attempting to authenticate to the read-only replica server in this environment, the user is only locked out from the read-only replica server when the incorrect password is specified three times. The read-only replica server cannot update the password policy operational attribute values for the entry on the master server. In this example, the user is still allowed to authenticate to the master server although the user is locked out on the replica server. To prevent these types of situations in

a replication environment, you should use peer-to-peer replication environments when password policy is configured and active. This enables all servers the ability to update the password policy operational attributes during authentication.

If you use read-only replicas in the advanced replication environment, then enable the read-only replica password policy replication support, also know as 'replication of bind failure on read-only replica'. With this support enabled, the read-only replica binds to the server specified by the **ibm-replicaReferralUrl** attribute in the replication context using the credentials that are passed to it, by the client. The supplier server pointed to by the **ibm-replicaReferralUrl** attribute updates the password policy operational attributes of the user on the supplier and then replicates those attributes to the read-only replica. If the credentials are incorrect, then a **pwdFailureTime** attribute value is added and the account might be locked, if the credentials are correct then all values for the **pwdFailureTime** attribute are removed. When these updates are replicated by the supplier server, the user entry has the same password policy state on all servers in the replication collection.

Note: Read-only replica password policy replication supports bind operations using simple authentication, and authentications using compare operations with the **userPassword** attribute. Bind operations that use DIGEST-MD5 or CRAM-MD5 authentication mechanisms, though supported for password policy, do not cause an equivalent update of password policy operational attributes on the master server, and continues to behave as if the read-only replica password policy replication support is disabled.

Read-only replica password policy replication is not supported in the basic replication environment. The server compatibility level must be 8 or greater to support read-only replica password policy replication.

Enabling read-only replica password policy replication

Use the **ldapdiff** utility to ensure that all user entries have the same password policy operational attribute values on all servers throughout the replication collection. Add the **ibm-slapdReplicateSecurityAttributes** attribute with a value of TRUE to the `cn=Replication, cn=configuration` entry on every server in the replication collection. If a read-only replica does not have the support enabled, it does not bind to the supplier server, therefore, no other servers in the collection know about the failure. If the supplier server does not have the support enabled, an extra value is added to the user entry and early account lock out occurs.

Disabling read-only replica password policy replication

Update the **ibm-slapdReplicateSecurityAttributes** attribute with a value of FALSE in the `cn=Replication, cn=configuration` entry on every server in the replication collection to disable the support. Pending binds from the read-only replica to the master are discarded when the support is disabled.

Password policy related extended operations

A set of extended operations are provided that allow an LDAP root administrator or an administrator with the appropriate authority the ability to query the effective password policy for a user or group and to query the status of a user's account. See Chapter 9, “Administrative group and roles,” on page 159 for more information about administrative role authority. The **ldapexop** utility is provided to call these password policy extended operations. Table 51 on page 385 summarizes

the extended operations including the **ldapexop** operation value. See “ldapexop utility” on page 255 for more information.

Table 51. Password policy extended operations

ldapexop operation	ldapexop description	Overview
acctstatus	**Account status** extended operation	This extended operation is used to query the status of a user entry that contains a **userPassword** value. The status returned is if the user's account is opened, locked by an administrator, or the user's password is expired. See “Account status” on page 695 for more information.
effectpwdpolicy	**Effective password policy** extended operation	This extended operation is used to query a user's or group's effective password policy entries and the effective password policy attribute values. See “Effective password policy” on page 705 for more information.

See “Effective password policy examples” on page 378 and “Account status extended operation example” on page 390 for examples on using these extended operations.

Overriding password policy and unlocking accounts

An LDAP root administrator or an administrator with the appropriate authority can override typical password policy behavior for specific user entries by modifying the password policy operational attributes. See Chapter 9, “Administrative group and roles,” on page 159 for more information about administrative role authority. This section shows examples of how the effective password policy is overridden for specific users.

An LDAP administrator can prevent the password for a specific account or user from expiring by setting the **pwdChangedTime** attribute value to a date far in the future. This example uses the **ldapmodify** utility to set the password expiration time to January 1, 2200 at midnight Coordinated Universal Time.

```
ldapmodify –D adminDn –w adminPw
dn: cn=user1,c=us
changetype: modify
replace: pwdChangedTime
pwdChangedTime: 220001010000000Z
```

An LDAP administrator can unlock an account, that is locked because of excessive login failures, by removing the **pwdAccountLockedTime** and **pwdFailureTime** attributes from the user entry. This example uses the **ldapmodify** utility to perform these modifications.

```
ldapmodify –D adminDn –w adminPw
dn: cn=user2,c=us
changetype: modify
delete: pwdAccountLockedTime
-
delete: pwdFailureTime
```

An LDAP administrator can unlock an account because the password has expired by setting the **pwdChangedTime** attribute to the current time and removing the **pwdExpirationWarned** and **pwdGraceUseTime** attributes. The **pwdChangedTime** attribute value is set to the current time to avoid the user's password from expiring immediately. This example uses the **ldapmodify** utility to unlock or unexpire the user's account by setting the **pwdChangedTime** attribute to the current time of June 1, 2010 at 1:00 Coordinated Universal Time.

```
ldapmodify –D adminDn –w adminPw
dn: cn=user3,c=us
changetype: modify
replace: pwdChangedTime
pwdChangedTime: 20100601010000Z
-
replace: pwdExpirationWarned
-
replace: pwdGraceUseTime
```

An LDAP administrator can bypass forcing a user to change the password value after a password reset by removing the **pwdReset** attribute. This example uses the **ldapmodify** utility to remove the **pwdReset** attribute.

```
ldapmodify –D adminDn –w adminPw
dn: cn=user4,c=us
changetype: modify
delete: pwdReset
```

An LDAP administrator can force a user to change their password value by setting the **pwdReset** attribute value to true. This example uses the **ldapmodify** utility to set the **pwdReset** attribute value to true.

```
ldapmodify –D adminDn –w adminPw
dn: cn=user5,c=us
changetype: modify
replace: pwdReset
pwdReset: true
```

An LDAP administrator can administratively lock a user's account by setting the **ibm-pwdAccountLocked** operational attribute to true. This prevents the user from authenticating successfully to the LDAP server. This example uses the **ldapmodify** utility to set the **ibm-pwdAccountLocked** attribute value to true.

```
ldapmodify –D adminDn –w adminPw
dn: cn=user6,c=us
changetype: modify
replace: ibm-pwdAccountLocked
ibm-pwdAccountLocked: true
```

An LDAP administrator can administratively unlock a user's account by setting the **ibm-pwdAccountLocked** operational attribute to false. If a user's account is unlocked in this manner, it does not affect the state of the account with respect to being locked because of excessive password failures or an expired password.

```
ldapmodify –D adminDn –w adminPw
dn: cn=user7,c=us
changetype: modify
replace: ibm-pwdAccountLocked
ibm-pwdAccountLocked: false
```

If the **Server administration** server control is specified (the **-k** option in the **ldapmodify** utility) when modifying the **ibm-pwdAccountLocked** attribute from true to false, the **pwdAccountLockedTime** and **pwdFailureTime** attribute values are also automatically removed the user's entry. This removes the administrative lock and the lock from excessive password failures. However, it does not affect the state of the account for an expired password.

Unlocking or unexpiring the account of the LDAP root administrator (adminDN)

If the LDAP root administrator defined in the configuration file (**adminDN** configuration option) specifies an entry that resides in a CDBM, LDBM, or TDBM backend and the account becomes locked or expired under z/OS LDAP password policy rules, the root administrator is unable to authenticate to the server. If this occurs, the LDAP server **UNLOCK** modify command is available to unlock or unexpire the account of this root administrator.

To unlock the account of the LDAP root administrator defined in the configuration file (**adminDN** configuration option), use the LDAP server **UNLOCK** operator modify command. In this example, *dssrv* is the name of the LDAP server started task. If the command is entered from SDSF, it must be preceded by a slash (/):

```
F dssrv,UNLOCK ADMIN
```

If this command is successful, the account of the LDAP root administrator defined in the **adminDN** configuration option is unlocked or unexpired. The root administrator is then allowed to authenticate to the LDAP server at which time the password must be changed.

Password policy examples

This section contains examples of configuring global, group, and individual password policy entries and associating them with users and groups. This section also contains examples of using the **Effective password policy** and **Account status** extended operations.

Global password policy example

When the global password policy entry, **cn=pwdpolicy,cn=ibmpolicies**, is initially created in the CDBM backend, the policy is not enabled. This example uses the **ldapmodify** utility to activate the global password policy and to change its default values. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information about the **ldapmodify** utility.

```
ldapmodify -p port -D adminDn -w adminPw
dn: cn=pwdpolicy,cn=ibmpolicies
replace: x
ibm-pwdpolicy: true
pwdmaxage: 7776000
pwdexpirewarning: 5184000
pwdmaxfailure: 5
pwdlockout: true
pwdinhistory: 3
pwdminlength: 5
pwdchecksyntax: 1
```

After these modifications are made to the global password policy entry, the following policy is in effect for all existing entries that have **userPassword** attribute values:

- Passwords must be changed every 90 days (7776000 seconds) and password expiration warnings are sent on the **PasswordPolicy** response control starting 60 days (5184000 seconds) before the password expires.
- There are a maximum of five login failures before the user's account is locked and must be unlocked by an LDAP root administrator or an administrator with the appropriate authority. See Chapter 9, “Administrative group and roles,” on page 159 for more information about administrative role authority.

- The previous three password values are kept in the user's password history and the user is unable to reuse these password values.
- The new password value must have a minimum length of five characters.

Group password policy example

If there are a number of users that must have a password policy that differs from the global password policy, group password policies are used. These users must be placed in a static, dynamic, and nested group and those groups are updated to refer to a password policy entry. This example uses the **ldapadd** utility to add a password policy entry in the CDBM backend to be used as a group password policy. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information about the **ldapadd** client utility.

```
ldapadd -p port -D adminDn -w adminPw
dn: cn=group,cn=ibmpolicies
objectclass: pwdpolicy
objectclass: ibm-pwdpolicyext
objectclass: container
pwdminlength: 10
pwdinhistory: 5
pwdchecksyntax: 2
passwordminalphachars: 5
passwordminotherchars: 2
pwdmaxage: 5184000
pwdexpirewarning: 2592000
pwdattribute: userpassword
ibm-pwdpolicy: true
```

The characteristics are:

- Passwords must be changed every 60 days (5184000 seconds) and password expiration warnings are sent on the **PasswordPolicy** response control starting 30 days (2592000) before the password is to expire.
- The minimum length of password values is 10 characters, five must be alphabetic characters, and two must be non-alphabetic characters. Password syntax checking is enforced because the **pwdCheckSyntax** attribute is set to two.
- The previous five password values are kept in the user's password history and the user is unable to reuse these password values.

After the password policy entry is created, the group entry that must use this special password policy must be modified to set the **ibm-pwdGroupPolicyDN** operational attribute value. This example uses the **ldapmodify** utility to modify the existing `cn=group,c=us` entry to add an **ibm-pwdGroupPolicyDN** operational attribute value for the `cn=group,cn=ibmpolicies` password policy entry.

```
ldapmodify -p port -D adminDn -w adminPw
dn: cn=group,c=us
add: ibm-pwdgrouppolicydn
ibm-pwdgrouppolicydn: cn=group,cn=ibmpolicies
```

Although the `cn=group,cn=ibmpolicies` password policy entry created earlier is enabled by setting the **ibm-pwdPolicy** attribute value to true, the global password policy must be enabled to evaluate additional password policies (if it is not already). Set the **ibm-pwdGroupAndIndividualEnabled** attribute value to true in the global password policy entry. This example uses the **ldapmodify** utility to enable the evaluation of additional password policies in the LDAP server.

```
ldapmodify -p port -D adminDn -w adminPw
dn: cn=pwdpolicy,cn=ibmpolicies
replace: x
ibm-pwdgroupandindividualenabled: true
```

After the global password policy is enabled to evaluate additional password policies, users that are members of the cn=group,c=us group are subject to the password policy specified in the cn=group,cn=ibmpolicies entry.

Individual password policy example

If there are only a few users that must have a password policy that differs from the global password policy, an individual password policy can be created and used. The users that require this special password policy are updated to refer to a password policy entry.

This example uses the **ldapadd** utility to add a password policy entry in the CDBM backend to be used as an individual password policy.

```
ldapadd -p port -D adminDn -w adminPw
dn: cn=individual,cn=ibmpolicies
objectclass: pwdpolicy
objectclass: ibm-pwdpolicyext
objectclass: container
pwdminlength: 8
pwdgraceloginlimit: 3
pwdinhistory: 4
pwdchecksyntax: 1
pwdattribute: userpassword
ibm-pwdpolicy: true
```

The characteristics are:

- The minimum length of a password value is eight characters with no restrictions on alphabetic or numeric characters.
- The previous four password values are kept in the user's password history and the user is unable to reuse these password values.
- There are three grace logins allowed before the user's password expires.

After the password policy entry is created, the individual users needed to use this special password policy must be modified to set the **ibm-pwdIndividualPolicyDN** operational attribute value. This example uses the **ldapmodify** utility to modify the existing cn=user5,c=us entry to add an **ibm-pwdIndividualPasswordPolicyDN** operational attribute value for the cn=individual,cn=ibmpolicies password policy entry.

```
ldapmodify -p port -D adminDn -w adminPw
dn: cn=user5,c=us
add: ibm-pwdIndividualPolicydn
ibm-pwdIndividualPolicydn: cn=individual,cn=ibmpolicies
```

Although the cn=individual,cn=ibmpolicies password policy entry created earlier is enabled by setting the **ibm-pwdPolicy** attribute value to true, the global password policy must be enabled to evaluate additional password policies (if it is not already). Set the **ibm-pwdGroupAndIndividualEnabled** attribute value to true in the global password policy entry. This example uses the **ldapmodify** utility to enable the evaluation of additional password policies in the LDAP server.

```
ldapmodify -p port -D adminDn -w adminPw
dn: cn=pwdpolicy,cn=ibmpolicies
replace: x
ibm-pwdgroupandindividualenabled: true
```

After the global password policy is enabled to evaluate additional password policies, the cn=user5,c=us entry is subject to the password policy specified in the cn=individual,cn=ibmpolicies entry.

Effective password policy extended operation example

The **Effective password policy** extended operation in the **ldapexop** utility is used to query the effective password policy of a user or group. The **Effective password policy** extended operation displays the password policy attribute values and the password policy entries that have contributed to the effective password policy for the specified user or group. See "ldapexop utility" on page 255 for more information. This example uses the **Effective password policy** extended operation in the **ldapexop** utility to query the effective password policy for user `cn=user5,c=us`:

```
ldapexop -p port -D adminDn -w adminPw -op effectpwdpolicy -d "cn=user5,c=us"

The effective password policy is calculated based on the following entries:
cn=pwdpolicy,cn=ibmpolicies
cn=group,cn=ibmpolicies
cn=individual,cn=ibmpolicies

The effective password policy is:
ibm-pwdgroupandindividualenabled=TRUE
ibm-pwdpolicy=TRUE
ibm-pwdPolicyStartTime=20090808153021.4210567Z
passwordmaxconsecutiverepeatedchars=0
passwordmaxrepeatedchars=0
passwordminalphachars=0
passwordmindiffchars=0
passwordminotherchars=0
pwdallowuserchange=TRUE
pwdattribute=userpassword
pwdchecksyntax=2
pwdexpirewarning=2592000
pwdfailurecountinterval=0
pwdgraceloginlimit=3
pwdinhistory=4
pwdlockout=TRUE
pwdlockoutduration=0
pwdmaxage=5184000
pwdmaxfailure=5
pwdminage=0
pwdminlength=8
pwdmustchange=TRUE
pwdsafemodify=FALSE
```

Note:

1. Because `cn=user5,c=us` has an individual password policy (`cn=individual,cn=ibmpolicies`) and is a member of a group that has an activated group password policy (`cn=group,cn=ibmpolicies`), the effective password policy is calculated based on each of these password policy entries.
2. The **ibm-pwdPolicyStartTime** attribute value returned in the **Effective password policy** extended operation example is a result of the individual password policy start time because that policy is evaluated first.

Account status extended operation example

The **Account status** extended operation in the **ldapexop** utility is used to query if the user's account is opened, locked, or the password has expired.

```
ldapexop -p port -D adminDn -w adminPw -op acctstatus -d "cn=user1,c=us"
acctstatus_extended_op: Account is locked.
```

Changing password values when pwdsafemodify is set to true

If the **pwdSafeModify** attribute in the effective password policy is set to true, the current and new password value must be provided when changing a user's **userPassword** attribute value.

The **ldapchangepwd** utility is provided in z/OS and you must specify the current and new password values when modifying the **userPassword** attribute value. See

z/OS IBM Tivoli Directory Server Client Programming for z/OS for more information. This example uses the **ldapchangepwd** utility to change the **userPassword** attribute value for user `cn=user1,c=us` from `secret` to `supersecret`.

For example:

```
ldapchangepwd -p port -D cn=user1,c=us -w secret -n supersecret
```

If changing passwords with a non-z/OS LDAP client, use the **ldapmodify** utility to specify the current and new **userPassword** attribute values as shown in the example. This example changes the password for user `cn=user2,c=us` from `secret` to `supersecret`.

```
ldapmodify -p port -D cn=user2,c=us -w secret
dn: cn=user2,c=us
changetype: modify
delete: userpassword
userpassword: secret
-
add: userpassword
userpassword: supersecret
```

Chapter 19. Kerberos authentication

The 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 vendor-acquired, 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.

The LDAP server supports Kerberos integrity and confidentiality services. Upon successful completion of a SASL bind operation using the GSS API mechanism, the negotiated quality of protection (QOP) is used for subsequent messages sent over the connection. This QOP continues to be used until the completion of a new SASL bind request. If the new SASL bind request fails, the connection reverts to anonymous authentication with no integrity or confidentiality services.

Setting up for Kerberos

Kerberos Version 5 binds, defined in RFC 2222: *Simple Authentication and Security Layer (SASL)*, are performed using the Generic Security Services Application Programming Interface (GSS API) defined in RFC 2743: *Generic Security Service Application Program Interface Version 2, Update 1* and RFC 2744: *Generic Security Service API Version 2 : C-bindings* . 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 starts the LDAP server. For example:

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

   where *ldap_prefix* is either `ldap` or `LDAP` and *primary-dns-hostname* is the primary hostname for the system in DNS. The Kerberos principal name associated with the LDAP server can be *ldap_prefix*/*primary-dns-hostname*@*krbRealmName*. The *krbRealmName* is the Kerberos realm in which the LDAP server operates. Use `ldap` to assure interoperability with all LDAP clients. `LDAP` is accepted, but this value is not usable with many non-z/OS LDAP clients.

   Example:

   ```
   ALTUSER LDAPSRV PASSWORD(sec1ret) NOEXPIRED KERB(KERBNAME(ldap/test.server.ibm.com))
   ```

3. If the KDC (Key Distribution Center) is not located on the same machine as the LDAP server, you must generate a key table 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 KEY VERSION from the **LISTUSER** command:

```
keytab add ldap_prefix/primary-dns-hostname -p password -v key-version
```

The **-k** *filename* option might also be used if you want to use your own key table 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 key table. However, the user ID that starts the server must have at least read access to IRR.RUSERMAP in the FACILITY class when the **KRB5_SERVER_KEYTAB** environment variable in the security server configuration file (**krb5.conf**) is set to 1. 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
```

See *z/OS Integrated Security Services Network Authentication Service Administration* for more information about configuring the KDC.

4. Enable your configuration file for Kerberos authentication.

```
# Global Section
supportKrb5 on
serverKrbPrinc LDAP/myhost.com@MYREALM.COM
krbLDAPAdmin ibm-kn=ldapadm@MYREALM.COM
krbKeytab none
# TDBM Section
krbIdentityMap on
# LDBM Section
krbIdentityMap on
# SDBM Section
krbIdentityMap on
# CDBM Section
krbIdentityMap on
```

Note:

a. In the above example, `myhost.com` is the primary DNS host name of the LDAP server and `MYREALM.COM` is the Kerberos realm to which the LDAP server belongs.

b. The first portion of the **serverKrbPrinc** identity can either be `ldap` or `LDAP` in the server configuration file and in the Kerberos segment of the RACF ID where it is defined. Use `ldap` to assure interoperability with all LDAP clients. `LDAP` is accepted, but this value is not usable with many non-z/OS LDAP clients. Check your KDC for case requirements.

c. The **serverKrbPrinc** configuration option is optional if the Kerberos principal name (KERBNAME field in RACF) of the LDAP server's user ID is "ldap/*primary-dns-hostname*", where *primary-dns-hostname* is the primary DNS hostname of the LDAP server.

5. Start your server. Your LDAP server is now configured with Kerberos support.

Schema for Kerberos

The LDAP server schema always contains the schema elements needed for Kerberos GSS API Authentication. No additional schema is needed.

Table 52 lists the Kerberos object classes and attributes.

Table 52. 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 might 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 not case-sensitive Kerberos server, then the Kerberos identity associated with this entry is stored as an **altSecurityIdentities** 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 is used in access control lists. This is known as the default mapping and is always performed when a SASL bind with a mechanism of GSS API is performed.

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 are used for access control throughout the server.

TDBM, LDBM, and CDBM mapping

The name specified in the BIND request must match the DN for the source principal used to establish the GSS API context, one of the mapped Kerberos identities, or must be null.

The DN for the source principal is formed as `ibm-kn=`*principal@REALM* where *principal@REALM* is obtained from the GSS API client credentials. Additional distinguished names can be assigned using Kerberos identity mapping. Kerberos identity mapping for a backend is enabled by setting the **krbIdentityMap** configuration option to **on**.

When Kerberos identity mapping is enabled for a backend, the following steps are performed:

1. Search each naming context managed by the backend for an entry with **objectClass=krbRealm-V2** and **krbRealmName-V2=*REALM***, where *REALM* is the realm portion of the bound Kerberos identity. If an entry is found, the **krbPrincSubtree** attribute values for the entry specify the directory subtrees to search in the next step.
2. Search each directory subtree specified by a **krbPrincSubtree** attribute value for entries containing a **krbPrincipalName** attribute with the same value as the bound Kerberos identity. The DN of each matching entry is added to the list of mapped Kerberos identities. If a matching entry also contains the **krbAliasedObjectName** attribute and the aliased entry specified by the **krbAliasedObjectName** attribute contains a **krbHintAliases** attribute value matching the name of the matching entry, then the DN of the aliased entry is also added to the list of mapped Kerberos identities.
3. Search each naming context managed by the backend for entries with **objectClass=ibm-securityIdentities** and either a **userPrincipalName** attribute with the value *principal@realm* or an **altSecurityIdentities** attribute with the value *kerberos:principal@REALM*. *kerberos* is any mixed-case spelling of `KERBEROS`. The DN of each matching entry is added to the list of mapped Kerberos identities.

Alias and referral search processing is not performed during Kerberos identity mapping.

SDBM mapping

If an SDBM backend is configured in the LDAP server configuration file and the **krbIdentityMap** configuration option is **on**, then the SDBM backend tries to map the Kerberos identity to the appropriate RACF ID. SDBM checks that the realm specified in the Kerberos identity is the RACF local realm and searches for a RACF user whose KERBNAME value in the KERB segment is the same as the principal in the Kerberos identity. If a RACF ID is found, then the SDBM DN that represents the RACF ID is added to the list of DNs. In this case, the bound user can then perform SDBM operations to access RACF data, under the context of the RACF ID.

Configuring access control

Since there is now a list of alternate DNs, access control is changed to operate on the list of DNs rather than just a single DN. Group gathering is also performed on all 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 is mapped to `ibm-kn=jeff@IBM.COM` and he would get read access to normal attributes 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-V2
      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 entry 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` 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 access to the `cn=Ken,o=IBM,c=US` entry since TDBM or LDBM 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 entry 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, it is necessary to look at the Tim entry before adding `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** value, 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` still has 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, LDBM, or CDBM mapping was performed and Jeff was aliased to Tim.

3. Use existing ACLs (Method 2). This method is used for not case-sensitive KDCs. Set up your TDBM, LDBM, or CDBM entries with the **altSecurityIdentities** attribute.

   Example:

   ```
   dn: cn=Jeff,o=IBM,c=US
   objectclass: ibm-securityIdentities
   altSecurityIdentities: KERBEROS:jeff@IBM.COM
   ```

   Now if `jeff@IBM.COM` performs a Kerberos bind he is mapped to `ibm-kn=jeff@IBM.COM` and `cn=Jeff,o=IBM,c=US`.

   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, LDBM, or CDBM 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: Because of space limitations in the diagram, the entries in the example do not contain all 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.

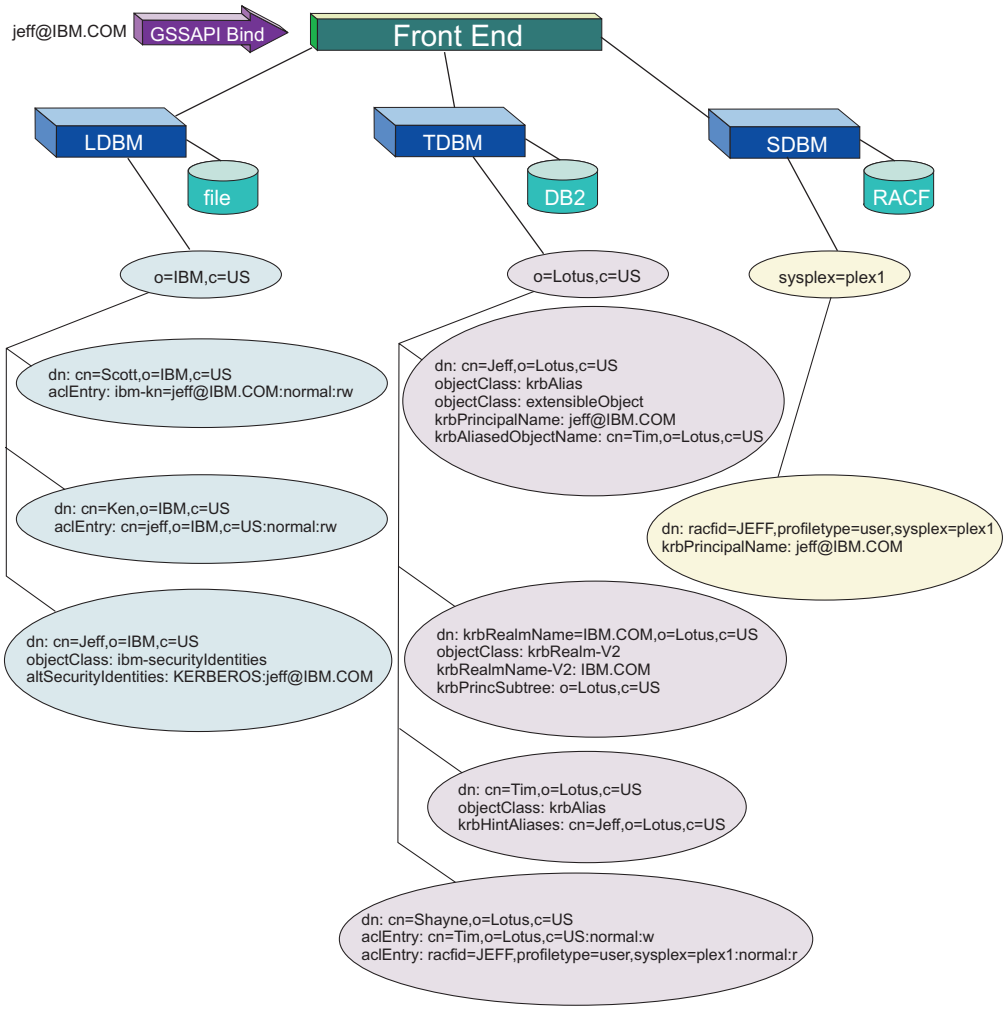


Figure 26. Kerberos directory example

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 a TDBM backend, an LDBM backend, 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 LDBM backend first looks for the Kerberos realm object with a `krbRealmName-V2=IBM.COM` and does 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 backend and tries to find the Kerberos realm object with a krbRealmName-V2=IBM.COM. This time the realm object is found so all 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 must 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 be an alias, 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
```

See "Associating DNs, access groups, and additional bind and directory entry access information with a bound user" on page 457 for information about group gathering after a successful bind.

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 attributes in the cn=Scott,o=IBM,c=US entry.
- jeff@IBM.COM also has read and write permission to the normal attributes in the cn=Ken,o=IBM,c=US entry because of 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 various 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 store Kerberos identities. In this scenario, a trusted realm must 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**, see *z/OS Integrated Security Services Network Authentication Service Administration*.

Chapter 20. Native authentication

The z/OS LDAP server has the ability to authenticate to the Security Server through the TDBM, LDBM, or CDBM backends by specifying a Security Server password or password phrase on a simple bind to the 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** or **uid** attribute to specify the Security Server ID that is associated with this entry. The ID and password or password phrase are passed to the Security Server and the verification of the password or password phrase is performed by the Security Server. Another feature of native authentication is the ability to change your password or password phrase on the Security Server by issuing an LDAP modify command.

Note:

1. The SDBM backend does not have to be configured in order to use native authentication.
2. After a successful native authentication bind, the bound user can send LDAP requests to any of the configured backends. If SDBM is configured, SDBM operations are performed under the context of the Security Server ID that was used during the native authentication bind. For all other backends, LDAP operations are performed using the normal bind information (the bind DN and the groups to which it belongs).
3. The use of RACF passtickets is supported by the z/OS LDAP server when using native authentication. The job name associated with the LDAP server started task should be used as the application name when generating RACF passtickets. See *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 with an LDBM, TDBM, or CDBM backend and then start the server. Specify the native authentication options in your LDAP server configuration file. For example:

```
# TDBM Section
useNativeAuth SELECTED
nativeAuthSubtree o=IBM,c=US
nativeAuthSubtree o=Lotus,c=US
nativeUpdateAllowed ON
```

3. 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.

Schema for native authentication

The LDAP server schema always contains the schema elements needed for native authentication. No additional schema is needed.

Following is the native authentication attribute type:

ibm-nativeId
Specifies the Security Server ID that is to be associated with this entry.

Following is the native authentication object class:

ibm-nativeAuthentication
Allows specifying the **ibm-nativeId** attribute in entries.

Defining participation in native authentication

There are many different configuration options for native authentication which are mentioned 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 are subject to native authentication.

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 or password phrase modifications, then you must list all the subtrees with multiple **nativeAuthSubtree** configuration options.

Note: If the DN that is listed in the **nativeAuthSubtree** options contains a space character in it, then the entire DN must be enclosed in quotes in the LDAP server configuration file.

In order for an entry to bind natively or perform a native password or password phrase 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. 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 user ID specified by either the **uid** or **ibm-nativeId** attributes must contain a valid OMVS segment with an OMVS UID value in the Security Server. If a native entry has an existing **userPassword** attribute value because it was originally created under a non-native authentication subtree and the Security Server identity that is specified has not yet been defined in the Security Server, the LDAP server attempts an LDAP simple bind. Similarly, if a Security Server identity is defined but it does not contain an OMVS segment, the LDAP server attempts an LDAP simple bind.

If you use the **useNativeAuth** option, also specify the **nativeUpdateAllowed** option to enable native password or password phrase changes in the Security Server to occur through the TDBM, LDBM, or CDBM backend.

An entry that is participating in native authentication cannot normally contain the **userPassword** attribute. An LDAP add request of an entry that contains a **userPassword** attribute value fails. An LDAP modify request that enables an entry for native authentication removes any existing **userPassword** attribute values for the entry.

Binding with native authentication

As mentioned above, there are two LDAP operations affected: bind and password or password phrase 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 binding.

Table 53. Operating modes for native authentication binding

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.
Bind	all	any value			For ease of implementation, an LDAP simple bind is attempted, even though you have specified that all entries should use native authentication. This entry should be configured correctly.
Notes: This table assumes that the entry is located within native authentication subtrees.					

In native authentication binding, the LDAP server invokes the RACROUTE REQUEST=VERIFY, ENVIR=CREATE macro using the mapped user ID and the password or password phrase supplied in the bind request. The following LDAP reason codes are mapped to return codes returned by the RACROUTE REQUEST=VERIFY,ENVIR=CREATE macro:

Table 54. LDAP return and reason codes returned to the client when binding with native authentication

LDAP return code	Reason code	Text
LDAP_INVALID_CREDENTIALS	R004111	The password is not correct
LDAP_INVALID_CREDENTIALS	R004112	A bind argument is not valid
LDAP_INVALID_CREDENTIALS	R004109	The password has expired

Table 54. LDAP return and reason codes returned to the client when binding with native authentication (continued)

LDAP return code	Reason code	Text
LDAP_INVALID_CREDENTIALS	R004128	Native authentication password change failed: The new password is not valid, or does not meet requirements
LDAP_INVALID_CREDENTIALS	R004110	The user ID has been revoked
LDAP_OPERATIONS_ERROR	R000208	Unexpected racroute error safRC=*safRC* racfRC=*racfRC* racfReason=*racfReason*

Note: The same reason codes are issued when binding with a password or a password phrase.

Updating native passwords and password phrases

To update a native password or password phrase, the LDAP server invokes the RACROUTE REQUEST=VERIFY, ENVIR=CREATE macro using the mapped user ID and the old and new passwords or password phrases supplied in the modify delete/add request. The following LDAP reason codes are mapped to return codes returned by the RACROUTE REQUEST=VERIFY,ENVIR=CREATE macro:

Table 55. LDAP return and reason codes returned to the client when updating the password or password phrase

LDAP return code	Reason code	Text
LDAP_INVALID_CREDENTIALS	R004111	The password is not correct
LDAP_INVALID_CREDENTIALS	R004112	A bind argument is not valid
LDAP_INVALID_CREDENTIALS	R004109	The password has expired
LDAP_INVALID_CREDENTIALS	R004128	Native authentication password change failed: The new password is not valid, or does not meet requirements
LDAP_INVALID_CREDENTIALS	R004110	The user ID has been revoked
LDAP_OPERATIONS_ERROR	R004118	Unexpected racroute error safRC=*safRC* racfRC=*racfRC* racfReason=*racfReason*

Note: The same reason codes are issued when updating a password or a password phrase.

Updating native passwords or password phrases during bind

Note: This section applies only to changing native passwords during bind. This method cannot be used to change the **userPassword** value during a bind to a TDBM, LDBM, or CDBM entry that does not use native authentication.

It is also possible to change the RACF password or password phrase of a TDBM, LDBM, or CDBM entry participating in native authentication during an LDAP simple bind. This may be necessary if the **ldapmodify** command above fails with LDAP return code **LDAP_INVALID_CREDENTIALS** and LDAP reason code:

```
R004109 The password has expired
```

The simple bind occurs as part of an LDAP function such as search, compare, add, or modify. The password or password phrase change is provided in the password portion of the LDAP simple bind. The change must be in the following format:

```
currentvalue/newvalue
```

The current value and the new value must both be passwords or both be password phrases. An error is returned if one of the values is a password and the other is a password phrase.

The forward slash (/) is used as the indication of a password or password phrase change during the LDAP simple bind. Password or password phrase changes made using the LDAP simple bind to a TDBM, LDBM, or CDBM entry participating in native authentication are subject to the system password or password phrase rules. A password or password phrase change fails 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 or password phrase does not pass the rules established on the system.

Note: A forward slash (/) is a legal character in a password phrase (but not in a password). During native authentication bind, a backward slash (\) is an escape character to indicate the next character is part of the password or password phrase and has no special meaning. The backward slash is removed during bind processing. Therefore, during bind, a forward slash in a password phrase must be preceded by a backward slash (\) to indicate that the forward slash is part of the password phrase and is not the password phrase change indicator. For example, the password phrase `this1slash/ispartofthevalue2use` must be specified as `this1slash\/ispartofthevalue2use` during bind. A backward slash is a legal character in a password phrase (but not in a password). Therefore, a backward slash in a password phrase must be preceded by another backward slash to indicate that it is not an escape character.

Once the bind succeeds, the password or password phrase is changed even if the LDAP function eventually fails. The **nativeUpdateAllowed** server configuration option setting does not control whether password or password phrase modifications can occur on an LDAP bind operation. The setting of **nativeUpdateAllowed** only controls password or password phrase modifications on an LDAP modify operation.

Assuming an LDBM or TDBM entry `cn=USER1,ou=END,o=IBM,c=US` is participating in native authentication and is mapped to user ID `USER1`, 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=*"
```

Password policy with native authentication

When authenticating with a user in an LDBM, TDBM, or CDBM backend that is participating in native authentication or doing the special delete-add modification of the bound user's **userPassword** attribute value, the password policy applied is determined by the underlying z/OS Security Server. Therefore, any configured z/OS LDAP password policy does not apply in these scenarios.

When the **PasswordPolicy** control is sent on a bind or modify request, the **PasswordPolicy** response control is returned on the bind and modify response and has additional warning and error information about the authenticating user's password or the updating of the native password or password phrase with the special delete-add modification operation. Based on information returned from the Security Server, only the following **PasswordPolicy** response control error codes

are supported: **accountLocked, changeAfterReset, insufficientPasswordQuality, mustSupplyOldPassword, passwordExpired**, and **passwordModNotAllowed**. See "PasswordPolicy" on page 684 for more information.

Also, note that a native authentication bind using an expired native password succeeds, if the **PasswordPolicy** control is included in the bind request and the **nativeUpdateAllowed reset** configuration file option is specified. After the bind, only the special delete-add modification of the bound user's **userPassword** attribute can be performed to reset the native authentication password. After completion, other LDAP operations can be performed.

Example of setting up native authentication

The following diagram shows an example of how you could set up native authentication.

Note: Because of space limitations in 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.

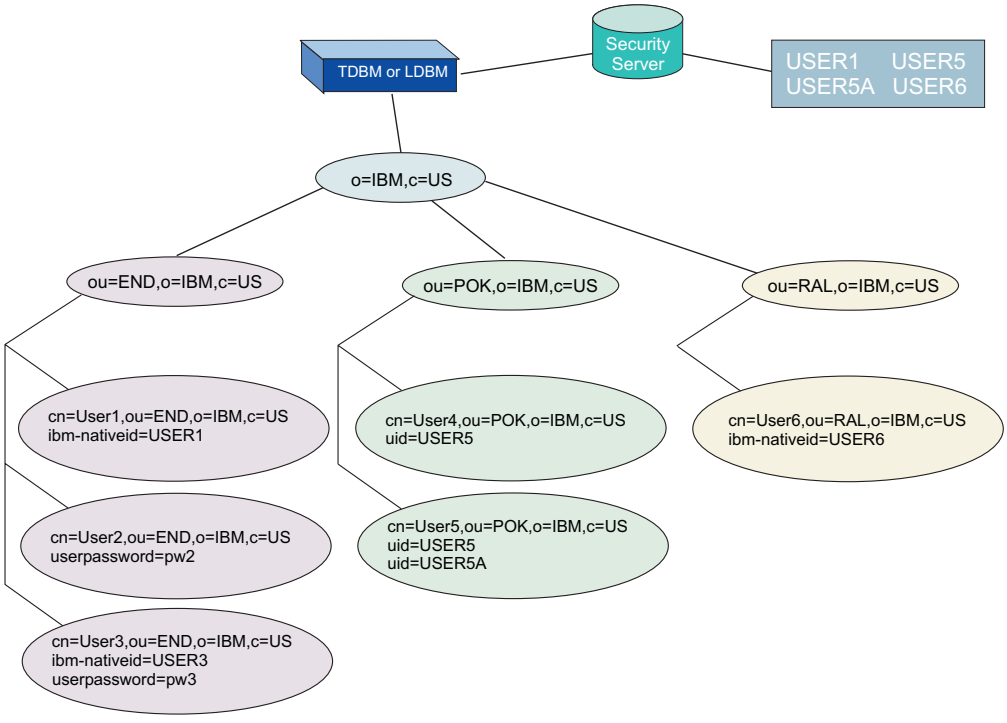


Figure 27. Native authentication example

Note: In the behavior table for each of the following examples, a password phrase can be used instead of a password, with the same results.

Example 1:

- Assuming these settings:
  useNativeAuth selected
  nativeUpdateAllowed on
  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 56. 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**.
	Bind with native password change	Can change this native password because the entry contains a valid **ibm-nativeId**.
	modify-delete and modify-add (**userPassword**)	Can change this native password because the entry contains a valid **ibm-nativeId**.
	modify-replace (**userPassword**)	Cannot perform a modify-replace of the **userPassword** attribute 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 fails because the Security Server ID USER3 is not defined, then a regular LDAP bind is performed.
	Bind with native password change	Cannot change the password on the bind because the Security Server ID USER3 is not defined.
	modify-delete and modify-add (**userPassword**)	Native password change is attempted but fails because the Security Server ID USER3 is not defined.
	modify-replace (**userPassword**)	An attempt to modify-replace the **userPassword** attribute fails 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:

- Assume these settings:
  useNativeAuth all
  nativeUpdateAllowed on
  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 57. 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**.
	Bind with native password change	Can change this native password because the entry contains a valid **ibm-nativeId**.
	modify-delete and modify-add (**userPassword**)	Can change this native password because the entry contains a valid **ibm-nativeId**.

Table 57. Behavior of native authentication in example 2 (continued)

LDAP Entry	Operation	Behavior
	modify-replace (**userPassword**)	Cannot perform a modify-replace of the **userPasssword** attribute 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.
	Bind with native password change	Cannot change the password on the bind because the entry is not properly set up for native authentication. A regular LDAP bind is attempted.
	modify-delete and modify-add (**userPassword**)	Because there are no native attributes on this entry, native authentication password update is not attempted. A regular modification of the **userPassword** attribute value is attempted.
	modify-replace (**userPassword**)	Because there are no native attributes on this entry, native authentication password update is not attempted. A regular modification of the **userPassword** attribute value is attempted.
cn=User3,ou=END,o=IBM,c=US	Bind	Attempts native authentication but fails because the Security Server ID USER3 is not defined, then a regular LDAP bind is performed.
	Bind with native password change	Cannot change the native password on the bind because the Security Server ID USER3 is not defined.
	modify-delete and modify-add (**userPassword**)	Native password change is attempted but fails because the Security Server ID USER3 is not defined.
	modify-replace (**userPassword**)	An attempt to modify-replace the **userPassword** attribute fails because the entry is configured for native authentication.
cn=User4,ou=POK,o=IBM,c=US	Bind	Can bind natively because the entry contains a valid **uid** (with one value).
	Bind with native password change	Can change this native password because the entry contains a valid **uid** (with one value).
	modify-delete and modify-add (**userPassword**)	Can change this native password because the entry contains a valid **uid** (with one value).
	modify-replace (**userPassword**)	An attempt to modify-replace the **userPassword** attribute fails 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.
	Bind with native password change	Cannot change the native password on the bind because 2 **uid** attribute values exist.
	modify-delete and modify-add (**userPassword**)	Cannot change the native password on modify operations because 2 **uid** attribute values exist.
	modify-replace (**userPassword**)	An attempt to modify-replace the **userPassword** fails because the entry is configured for native authentication.
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 or password phrase (from the system where the LDAP server is running). The web server searches the LDAP directory for an entry where **uid** equals the input user ID. The web server uses the returned DN and the inputted password or password phrase to do an **ldap_simple_bind()**. When the LDAP server determines this entry is subject to native authentication, it retrieves the **ibm-nativeId** or **uid** value and verify the password or password phrase with the Security Server. Note that if **useNativeAuth** is set to **selected**, it might 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 21. CRAM-MD5 and DIGEST-MD5 authentication

The z/OS LDAP server allows clients to authenticate using the CRAM-MD5 (Challenge Response Authentication Mechanism) and DIGEST-MD5 SASL bind mechanisms. CRAM-MD5 is defined in RFC 2195: *IMAP/POP AUTHorize Extension for Simple Challenge/Response*. DIGEST-MD5 is defined in RFC 2831: *Using Digest Authentication as a SASL Mechanism*. Both the CRAM-MD5 and DIGEST-MD5 mechanisms are multi-stage binds where the server sends the client a challenge and then the client sends a challenge response back to the server to complete the authentication. The client challenge response contains a hash of the password entered by the user, the username, and other pieces of data encoded to the specifications of either the CRAM-MD5 or DIGEST-MD5 RFCs.

The CRAM-MD5 and DIGEST-MD5 SASL bind mechanisms are more secure than performing simple binds since the credentials are not passed in clear text. Also, the CRAM-MD5 and DIGEST-MD5 bind mechanisms on the z/OS LDAP server do not require any additional products to be installed or configured.

The z/OS LDAP server DIGEST-MD5 bind mechanism supports the integrity and confidentiality options defined in RFC 2831: *Using Digest Authentication as a SASL Mechanism*. Upon the successful completion of a DIGEST-MD5 bind, the negotiated quality of protection (qop) is used for subsequent messages sent over the connection. The negotiated qop continues until the completion of a new SASL bind request. If the new SASL bind request fails, the connection reverts to anonymous authentication with no integrity or confidentiality support.

The DIGEST-MD5 authentication mechanism is more secure than the CRAM-MD5 authentication mechanism because it prevents chosen plaintext password attacks. During a DIGEST-MD5 authentication exchange between a client and the server, there is additional information passed which is used to construct a more robust hashing algorithm when compared against a CRAM-MD5 authentication making it more difficult to decipher.

DIGEST-MD5 bind mechanism restrictions in the z/OS LDAP server

DIGEST-MD5 restrictions on the LDAP server:

1. 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.
2. Subsequent authentication is not supported.

Considerations for setting up a TDBM, LDBM, or CDBM backend for CRAM-MD5 and DIGEST-MD5 authentication

The following are considerations for setting up a TDBM, LDBM, or CDBM backend for CRAM-MD5 and DIGEST-MD5 authentication:

1. In order to use the CRAM-MD5 bind mechanism on the LDAP server, the TDBM, LDBM, or CDBM entries that you bind with should contain a **uid** attribute value. The **uid** attribute type is always present in the minimum schema of the server. There are three ways to perform a CRAM-MD5 bind to the LDAP server:

a. Only specifying the bindDN in the bind request in your client application. When using the z/OS LDAP client utilities, such as **ldapsearch**, this is done by only specifying the **-D** option.

b. Only specifying the username in the CRAM-MD5 bind mechanism in your client application. The username that is specified must map to one of the **uid** attribute values in one of the TDBM, LDBM, or CDBM entries. When using the z/OS LDAP client utilities, such as **ldapsearch**, this is done by only specifying the **-U** option.

c. Specifying both the bindDN in the bind request and the username in the CRAM-MD5 bind mechanism in your client application. The username that is specified must map to one of the **uid** attribute values in one of the TDBM, LDBM, or CDBM entries. The bindDN specified in the bind request must map to the same distinguished name as the username. When using the LDAP client utilities, such as **ldapsearch**, this is done by specifying both the **-D** and the **-U** options.

For more information about the z/OS LDAP client utilities, see *z/OS IBM Tivoli Directory Server Client Programming for z/OS*.

Assuming that the password entered on the client application is correct, the CRAM-MD5 bind is successful, otherwise it returns an LDAP credentials error.

2. In order to use the DIGEST-MD5 bind mechanism on the z/OS LDAP server, the TDBM, LDBM, or CDBM entries that you bind with must contain a **uid** attribute value. The **uid** attribute type is always present in the minimum schema of the server. There are two ways to perform a DIGEST-MD5 bind to the z/OS LDAP server:

   a. Only specifying the username in the DIGEST-MD5 bind mechanism in your client application. The username that is specified must map to one of the **uid** attribute values in one of the TDBM, LDBM, or CDBM entries. When using the z/OS LDAP client utilities, such as **ldapsearch**, this is done by only specifying the **-U** option.

   b. Specifying both the username and the authorization DN in the DIGEST-MD5 bind mechanism in your client application. The username that is specified must map to one of the **uid** attribute values in one of the TDBM, LDBM, or CDBM entries. The authorization DN that is specified must map to the same distinguished name as the username. When using the z/OS LDAP client utilities, such as **ldapsearch**, this is done by specifying both the **-D** and the **-U** options.

   For more information about the z/OS LDAP client utilities, see *z/OS IBM Tivoli Directory Server Client Programming for z/OS*.

   Assuming that the password entered on the client application is correct, the DIGEST-MD5 bind will be successful, otherwise it will return an LDAP credentials error.

3. It is strongly suggested that the **uid** attribute values specified on the entries to be used for CRAM-MD5 or DIGEST-MD5 authentication be unique across every TDBM, LDBM, and CDBM backend that is configured on the LDAP server. Authentication can fail if more than one entry has the same **uid** attribute value.

4. In order for the CRAM-MD5 and DIGEST-MD5 binds to work properly, the **userPassword** attribute values for the entry must be in clear text (not suggested) or encrypted in either DES or AES. DES and AES encryption are suggested since they both encrypt the **userPassword** and provide clear text decryption. For additional information about AES and DES encryption, see "Configuring for encryption or hashing" on page 77. The z/OS LDAP server needs access to the clear text password so that the CRAM-MD5 and DIGEST-MD5 bind mechanisms work properly against that entry.

5. CRAM-MD5 and DIGEST-MD5 binds are not supported with entries that are participating in native authentication.
6. CRAM-MD5 and DIGEST-MD5 binds are not supported to the SDBM backend.
7. LDAP password policy supports CRAM-MD5 and DIGEST-MD5 binds.

CRAM-MD5 and DIGEST-MD5 configuration option

The **digestRealm** option in the LDAP server configuration file 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 option 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 what you want. See the **digestRealm** option at "digestRealm hostname " on page 99. If the **digestRealm** configuration option is not specified, the realm name defaults to the fully qualified hostname of the system where the LDAP server is running assuming that a DNS (Domain Name Server) is available. If the **digestRealm** option is not specified and the fully qualified hostname of the LDAP server cannot be determined because of a problem with the DNS (Domain Name Server), any CRAM-MD5 or DIGEST-MD5 binds attempted will fail.

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 or LDBM backend.

Note: Because of space limitations in 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.

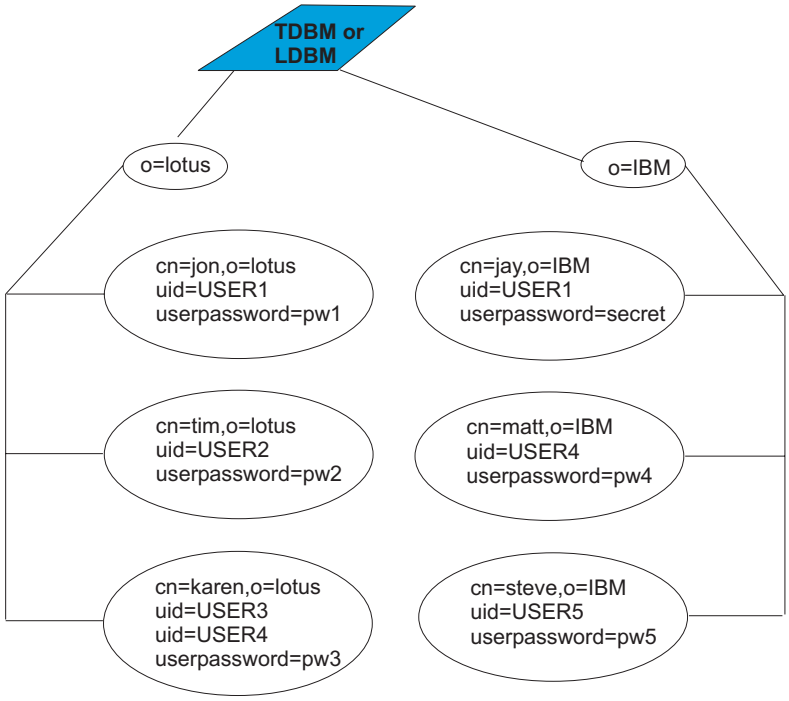


Figure 28. CRAM-MD5 and DIGEST-MD5 authentication example

The following table outlines what happens if you attempt to do a CRAM-MD5 or DIGEST-MD5 bind from a client. The username refers to the **-U** option on the z/OS LDAP client utilities, while the bindDN (CRAM-MD5) or authorization DN

(DIGEST-MD5) is the **-D** option on the z/OS LDAP client utilities. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more details on the LDAP client utilities. This table assumes that native authentication is not turned on under the subtrees: `o=lotus` and `o=IBM`.

Table 58. Behavior of CRAM-MD5 and DIGEST-MD5 authentication in example

Username	BindDN (CRAM-MD5) or authorization DN (DIGEST-MD5)	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 bindDN (CRAM-MD5) or authorization DN (DIGEST-MD5) `cn=jon,o=lotus` does not equal the username DN `cn=tim,o=lotus`
USER1		pw1	Bind is not successful, because there are multiple entries with the same username 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 username DN `cn=jay,o=IBM` equals the bindDN (CRAM-MD5) or authorization DN (DIGEST-MD5) `cn=jay,o=IBM`
USER1	`cn=jon,o=lotus`	pw1	Bind is successful to `cn=jon,o=lotus` because the username DN `cn=jon,o=lotus` equals the bindDN (CRAM-MD5) or authorization DN (DIGEST-MD5) `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 username DN `cn=karen,o=lotus` and bindDN (CRAM-MD5) or authorization DN (DIGEST-MD5) `cn=karen,o=lotus` because the password is incorrect.
USER5	`cn=nothere,o=lotus`	pw5	Bind is not successful because the username DN `cn=steve,o=IBM` does not equal the non-existent bindDN (CRAM-MD5) or authorization DN (DIGEST-MD5) `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 or LDBM backend.

Chapter 22. Using extended operations to access Policy Director data

The use of the extended operations (EXOP) backend, the **GetDnForUserid** and **GetPrivileges** extended operations, and the **IBMLdapProxyControl** are deprecated.

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. 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.

"GetDnForUserid" on page 706 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. See “GetPrivileges” on page 710 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.

“GetPrivileges” on page 710 summarizes some different error scenarios for this extended operation and the EXOP backend's response to such scenarios.

Chapter 23. Static, dynamic, and nested groups

The 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 24, "Using access control," on page 433 for additional information about access control checking.

The LDAP server supports static, dynamic, and nested groups. It is 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 enumerates all of the members that belong in that group. For a given user entry, the **ibm-allGroups** attribute determines the groups that the user has membership in.

A search request specifying the **ibm-allMembers** or **ibm-allGroups** attribute returns group membership information for just the backend containing the base entry. Access checking is performed for the **member** and **uniqueMember** attributes when obtaining the group membership information. Additional access checking is performed on any of the attributes contained in a dynamic group URL search filter on the **memberURL** attribute. Access checking is not performed on the **memberURL** and **ibm-memberGroup** attributes themselves.

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 that is called **member** to define a list of distinguished names (DNs) that belong to the static group. The **groupOfUniqueNames** object class uses a multi-valued attribute called **uniqueMember** to define a list of distinguished names (DNs) that belong to the static group. The **uniqueMember** attribute type is treated as a distinguished name and not as a distinguished name with an optional unique identifier.

These attributes and object classes are always in the LDAP server schema. Except for the **groupOfNames** and **groupOfUniqueNames** object classes, they cannot be modified. The **groupOfNames** and **groupOfUniqueNames** object classes can be modified in limited ways, as described in "Changing the initial schema" on page 299. One modification you may consider making in these two object classes is to move the **member** or **uniqueMember** attribute from the MUST list to the MAY list. This allows static group entries using these object classes to be created without any members and also allow all the members to be deleted from existing entries.

A typical static group entry is as follows:

```
dn: cn=ldap_team_static,o=endicott
objectclass: groupOfNames
cn: ldap_team_static
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 always in the LDAP server schema and cannot be modified.

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.

Alias and referral entries are not processed during the group membership search.

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. The dynamic URL is not used if the base entry is not within the same backend as the dynamic group entry. 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 IBM Tivoli Directory Server Client Programming for z/OS* for additional information about 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, which is defined in RFC 2255: *The LDAP URL Format* (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*. An add or modify operation of a dynamic group entry fails if it contains a **memberURL** attribute that is not in the correct format. This prevents introducing an improperly formatted **memberURL** attribute into the LDAP server. If migrating from an Integrated Security Services LDAP server on earlier releases and there are **memberURL** attribute values that not properly formatted in the directory, they are ignored by the LDAP server.

An entry is considered to be a member of the dynamic group if it falls within the search scope and matches the search filter. Alias entries and referral entries are

treated as normal entries during the group membership search; no alias dereferencing or referral processing is performed.

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 that is called **ibm-memberGroup** to indicate the DNs of the groups that are referenced by the nested group. This object class and attribute are always in the LDAP server schema and cannot be modified. Nested groups allow LDAP administrators to construct and display group hierarchies that describe both direct and indirect group memberships. A group that is referenced within the nested group is ignored if it is not in the same backend as the nested group. The group hierarchy that is established by a nested group cannot loop back to itself. The LDBM or CDBM backend rejects an add or modify operation of a nested group entry if it results in a loop. To be compatible with TDBM in the Integrated Security Services LDAP server on previous releases, the TDBM backend allows such an add or modify operation of a nested group. When the nested group is expanded, such as in an **ibm-allMembers** search of the group, TDBM detects the loop and continues with the next part of the expansion.

Note: The **ibm-nestedGroup** object class is an **AUXILARY** object class and also requires a **STRUCTURAL** object class.

A typical nested group entry is as follows:

```
dn: cn=ldap_team_nested,o=endicott
objectclass: container
objectclass: ibm-nestedGroup
```

```
cn: ldap_team_nested
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, LDBM, or CDBM backend that matches the scope and search filter contained in one of the values of the **memberURL** attribute of the group entry. Dynamic group membership is the union of all search expressions that are present on each of the individual **memberURL** attribute values even if the search expressions are contradictory, such as `ldap:///o=ibm??sub?cn=bob` and `ldap:///o=ibm??sub?(!(cn=bob))`. A dynamic search filter is ignored if the base in the search filter is not in the same TDBM, LDBM, or CDBM 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.

Zero-length values are ignored for the **member**, **uniqueMember** and **ibm-memberGroup** attributes.

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 within the same backend as the user entry to which that user has membership, including ancestor groups from nested group hierarchy. As with all operational attributes, they are only returned if explicitly requested and cannot be specified on a search filter.

The **ibm-allGroups** and **ibm-allMembers** search and comparison operations are only supported on entries within the TDBM, LDBM, or CDBM backend. These operations are not supported against users or groups that are present within the SDBM backend.

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. The entries and attributes that are used to evaluate **ibm-allMembers** and **ibm-allGroups** have ACL restrictions, against which the bound DN must be checked. 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, LDBM, or CDBM backend as the nested group.

Specifying **ibm-allMembers** or **ibm-allGroups** in a search or compare operation also requires that the bound DN have read 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 24, "Using access control," on page 433.

ACL restrictions on group gathering

If LDAP password policy is active, the list of the static, dynamic, and nested groups that the binding user is a member are gathered at authentication time. If LDAP password policy is not active, the list of static, dynamic, and nested groups are not gathered until the next non-bind request is received. 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 has to any of the attributes or subtrees in the directory until all the groups are fully determined.

Managing group search limits

Search operations can consume server resources and impair server performance. The LDAP server provides two configuration options, **sizeLimit** and **timeLimit**, that can be used to control the amount of server resources that are consumed during search operations. These options can limit the number of entries that a search returns and the duration of the search. The configuration limits apply to all users except administrators. However, some non-administrator users may need different search limits than those allowed by the configuration limits, either to allow them to perform larger searches or to limit them to smaller ones. Group search limits provide a way to override the server configuration search limits for groups of users.

Creating group search limits

A group search limit is created by adding all of the following to a static, dynamic, or nested group entry in an LDBM, TDBM, or CDBM backend:

- **ibm-searchLimits** object class
- **cn, ibm-searchSizeLimit,** and **ibm-searchTimeLimit** attributes

The object class and attributes are always in the LDAP server schema.

The **ibm-searchSizeLimit** and **ibm-searchTimeLimit** attributes are used to specify search limits for the members of the group. The value of each attribute can be:

-1 This group is not used to determine this limit.

0 This group of users has unlimited size or time.

1 - 2147483647
This group of users has the specified size or time (in seconds) limit.

Enabling group search limit processing

Groups containing group search limits can be created and modified at any time, but the limits in the groups are only used when the LDAP server is started with a server compatibility level of 7 or higher. See the **serverCompatLevel** configuration option at "serverCompatLevel {3 | 4 | 5 | 6 | 7| 8}" on page 125 for more information about the server compatibility level.

Using the limits from search limit groups

When a non-administrator user binds to the LDAP server and the server compatibility level is 7 or higher, the LDAP server determines all the groups containing group search limits among the groups in which the user belongs and then saves the highest **ibm-searchSizeLimit** and **ibm-searchTimeLimit** attribute values from these groups in the user's bind information. The highest values for size and for time might come from different groups in which the user belongs. Once the user's bind search limits are determined, they remain fixed until the user rebinds, even if the group search limits are later changed in the group entries or the user is added to or removed from groups with group search limits.

When the user issues a search request to the LDAP server, the size and time limits that are applied to the search are determined as follows:

1. The server first determines the server search limits. If the requester has a size or time search limit in the user bind information (from group search limits), that limit is used as the server limit. Otherwise, the server limit is set to the value specified by either the **sizeLimit** or the **timeLimit** configuration option.
2. The size and time limits that are used for the search are the smaller of the server search limit and the limits (if any) specified on the search request.

Note: Searches from administrators are not subject to any server limitations, either from configuration options or from group search limits.

Group examples

Examples of adding, modifying, and deleting group entries

Adding group entries: This example creates static group entries using the **accessGroup, groupOfUniqueNames,** and **groupOfNames** object classes. Group search limits are also specified in the first group to allow searches by group members to return up to 200,000 entries and take unlimited time.

```
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
objectclass: ibm-searchLimits
cn: group1
ibm-searchsizelimit: 200000
ibm-searchtimelimit: 0
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=Your Company??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`.

```
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: container
cn: nested_grp
ibm-memberGroup: cn=dynamic_team,o=Your Company
ibm-memberGroup: cn=group1,o=Your Company
```

Modifying group entries: 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
```

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
```

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)
```

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)
```

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
```

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: To delete a static, dynamic, and 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

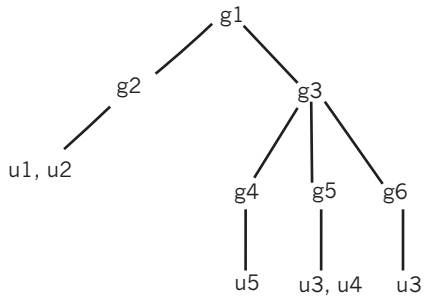


Figure 29. Group hierarchy and membership for the examples

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: ibm

dn: cn=g1,o=ibm
objectclass: container
objectclass: ibm-nestedGroup
cn: g1
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
cn: g2
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
cn: g3
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
cn: g4
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
cn: g5
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
cn: g6
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 secret6 –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 secret6 –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 cn=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 secret6 –b "cn=g3,o=ibm" "objectclass=*" ibm-allMembers

dn: cn=g3,o=ibm
ibm-allmembers: 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 entries, cn=u3,o=ibm and cn=g3,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 is not 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 secret3 -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 are 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 secret2 -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 is not 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 secret1 -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 is 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 secret3 "cn=g5,o=ibm" "ibm-allmembers=cn=u3,o=ibm"
ldap_compare: Compare true
```

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 secret6 -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 also. cn=g1,o=ibm has cn=g3,o=ibm as an **ibm-memberGroup** value, therefore cn=g1,o=ibm 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 secret1 -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 secret2 -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 secret3 -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 secret2 "cn=u3,o=ibm" "ibm-allGroups=cn=g1,o=ibm"
ldap_compare: Compare true
```

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 also. `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 returns an **LDAP_COMPARE_TRUE** to the client application.

Chapter 24. Using access control

Access control of information in the LDAP server is specified by setting up Access Control Lists (ACLs). LDBM, TDBM, CDBM, or GDBM ACLs provide a means to protect information that is stored in an LDAP directory. Administrators use ACLs to restrict access to different portions of the directory, or specific directory entries. When using the LDBM, TDBM, CDBM, or GDBM backend, ACLs are created and managed by using the **ldap_add** and **ldap_modify** APIs. ACLs can also be entered by using the **ldif2ds** utility (TDBM only).

ACLs are represented by a set of attributes that seem to be a part of the entry. The attributes that are 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 that 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 LDAPs SDBM backend allows a user to be authenticated to the directory namespace by 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 that are managed by the LDBM, TDBM, CDBM, or GDBM backend by using RACF-style user and group DNs. This controls access to LDBM, TDBM, CDBM, or GDBM database directory entries by using the RACF user or group identities.

The LDAP server schema entry also has an ACL that can be set to control access to the schema entry.

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 attributes that describe who is allowed to access information within that entry. Table 59 shows the set of attributes that are related to access control. More in-depth information about each attribute is given following the table.

It is possible to specify access control settings for individual attribute types. This is called attribute-level access control. Also, it is possible to explicitly deny access to information.

Table 59. ACL and entry owner attributes

ACL attributes	
aclEntry	This is a multi-valued attribute that contains the user or group distinguished names or search filters and permissions associated with those users or groups 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).

Table 59. ACL and entry owner attributes (continued)

ACL attributes	
aclPropagate	This is a single-valued boolean attribute that 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 managed by the LDAP server and cannot be changed by the **ldapmodify** utility. 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	
entryOwner	This is a multi-valued attribute that contains the user or group distinguished names or search filters that are evaluated 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 that 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 managed by the LDAP server and cannot be changed by the **ldapmodify** utility. 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

aclEntry is a multi-valued attribute that contains information pertaining to the access allowed to the entry and each of its attributes. **aclEntry** lists the following types of information:

- Who has rights to the entry (scope of the protection). Also called the subject.
- What specific attributes and classes of attributes (attribute access classes) that the subject has access to.
- What rights the subject has (permissions to specific attributes and classes of attributes).

Syntax

Following is the **aclEntry** attribute value syntax:

aclEntry: *aclEntry_value*[1]

1. where,

   aclEntry_value :- [**access-id**:|**group**:|**role**:]*subject_DN*[*granted_rights*]

   or,

subject_DN :- valid DN, the object that privileges are granted to.

filter :- valid search filter, using the following attributes only: **ibm-filterSubject, ibm-filterIP, ibm-filterTimeOfDay, ibm-filterDayOfWeek, ibm-filterBindMechanism**, and **ibm-filterConnectionEncrypted**. See "ACL filters" on page 441 for more information.

operation :- **union** | **replace** | **intersect**. See "ACL filters" on page 441 for more information.

granted_rights :- *object_rights* | *normal_rights* | *sensitive_rights* | *critical_rights* | *restricted_rights* | *system_rights* | *attr_rights*

object_rights :- **:object:**[**grant:** | **deny:**]*object_rights_list*

object_rights_list :- [**a** | **d**]

normal_rights :- **:normal:**[**grant:** | **deny:**]*attr_rights_list*

sensitive_rights :- **:sensitive:**[**grant:** | **deny:**]*attr_rights_list*

critical_rights :- **:critical:**[**grant:** | **deny:**]*attr_rights_list*

restricted_rights :- **:restricted:**[**grant:** | **deny:**]*attr_rights_list*

system_rights :- **:system:**[**grant:** | **deny:**]*attr_rights_list*

attr_rights :- :**at.***attr_name***:**[**grant:** | **deny:**]*attr_rights_list*

attr_name:- any valid attribute name

attr_rights_list :- [**r** | **w** | **s** | **c**]

The *subject_DN* is any valid DN that 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 [**a** | **d**], and *attr_rights_list* is one or more elements of the set [**r** | **w** | **s** | **c**].

See "ACL filters" on page 441 for more information about the *filter* and *operation* values.

Multiple specifications for the same access class or attribute type within the same **aclEntry** attribute value are merged into a single specification. For example:

```
group:cn=Anybody:normal:rs:system:rsc:normal:c:normal:deny:w
```

results this merged access list

```
group:cn=Anybody:normal:rsc:normal:deny:w:system:rsc
```

Scope of protection: The scope of the protection is based on the following types of privilege attributes:

access-id
The distinguished name of an entry to set permissions for.

group The distinguished name of the group entry to set permissions for.

role The distinguished name of the group entry to set permissions for.

aclFilter
The **aclEntry** filter, that if evaluates to true, reduces, augments, or replaces the set of permissions. The server compatibility must be 6 or greater to use

---

aclEntry_value :- **aclFilter:***filter*:*operation*[*granted_rights*]

an **aclFilter** in an **aclEntry** attribute. See "serverCompatLevel {3 | 4 | 5 | 6 | 7| 8}" on page 125 for more information about the server compatibility level.

Access control groups can be either static, dynamic, or nested groups. The following object classes are evaluated as group entries for the LDBM, TDBM, and CDBM backends: **ibm-staticGroup, groupOfNames, groupOfUniqueNames, accessRole, accessGroup, ibm-dynamicGroup, groupOfUrls,** and **ibm-nestedGroup**. See Chapter 23, "Static, dynamic, and nested groups," on page 419 for additional information about static, dynamic, and nested groups.

When specifying a user or group distinguished name in an **aclEntry** attribute value, the **access-id, group**, or **role** portions of the value are optional and are accepted for compatibility with older levels of the LDAP server. If replicating to non-z/OS IBM TDS, one of these prefixes are required. The distinguished name that is used does not need to be the name of any entry in the directory. The distinguished name is the name that represents the user that has authenticated to the directory server or the group that the user is a member of.

The access control implementation supports several "pseudo-DNs". These are used to reference 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 that is authenticated to the directory. The method of authentication is not considered when using distinguished names in the **aclEntry** attribute value.

The `access-id:cn=this` refers to the bind DN that matches the target object's DN on which the operation is performed.

If the server compatibility is 6 or greater, a search filter can be specified after the **aclFilter** component in an **aclEntry** attribute value. When the search filter evaluates to true, it reduces, augments, or replaces the set of permissions specified. See "ACL filters" on page 441 for more information about the search filters that are allowed to be specified.

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 a RACF identity that is established with SDBM in an ACL:

```
access-id:racfid=YourID,profileType=user,sysplex=YourSysplex
group:racfid=YourGrp,profileType=group,sysplex=YourSysplex
```

Attribute access classes

Attributes requiring similar permission for access are grouped in classes. Attributes are assigned to an attribute access class within the schema definitions. The **IBMAttributeTypes** attribute in the LDAP server schema entry holds the attribute types 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 that are 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, the name of a person is typically defined in the **normal** class. Perhaps a social security number is considered **sensitive**, and any password information for the user is considered **critical**. Following are some example definitions that are excerpted from the LDAP server schema. Note that the attribute **userPassword** is defined with access class **critical**.

```
attributetypes: (
  2.5.4.49
  NAME ( 'dn'  'distinguishedName'  )
  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 'Defines the user password'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
  USAGE userApplications
  )
ibmattributetypes: (
  2.5.4.35
  ACCESS-CLASS critical
  )
```

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.

Note: The **restricted** attributes are: **aclEntry**, **aclPropagate**, **entryOwner**, and **ownerPropagate**. 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. To update access control information, you must have permission to read and write

these attributes. If the **system** keyword is not specified in an **aclEntry** attribute value, the system access is set to 'system:rsc'.

Access permissions

Following is the set of access permissions.

Table 60. Permissions that apply to an entire entry

Permission	
Add	Add an entry below this entry
Delete	Delete this entry

Table 61. Permissions that apply to attribute access classes

Permission	
Read	Read attribute values
Write	Write attribute values
Search	Search filter can contain attribute type
Compare	Compare attribute values

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

group:cn=anybody:normal:rsc

group:cn=authenticated:normal:rwsc:sensitive:rsc

access-id:cn=this:normal:rwsc:sensitive:rwsc:restricted:rwsc

aclFilter: (&(ibm-filterTimeOfDay>=09:00)(ibm-filterTimeOfDay<=17:00)(ibm-filterDayOfWeek>=1)
 (ibm-filterDayOfWeek<=5)):union:normal:w

aclFilter: (|(ibm-filterSubject=cn=Ken, o=Your Company)(ibm-filterIP=129.176.132.*))
 :replace:normal:rwsc:critical:rsc:sensitive:rwsc
```

See “Access determination” on page 445 for information about how the **aclEntry** values are used to determine access.

The **aclEntry** attribute value is defined as a directory string.

When the **aclFilter** scope is not specified, a search using the **aclEntry** attribute matches against the distinguished name in the value. An **aclEntry** value in this format is normalized following the matching rules for a distinguished name. Two **aclEntry** attributes in this format are considered to be the same if they have the same distinguished name.

When the **aclFilter** scope is specified, a search using the **aclEntry** attribute matches against the scope, filter, and operation in the search filter. An **aclEntry** in this

format is normalized by normalizing the scope, filter, and the operation. Two **aclEntry** attributes in this format are considered to be the same if they have the same filter and operation.

aclPropagate attribute

An **aclPropagate** attribute is associated with each entry with an explicit ACL. By default, the entrys 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 455 for more information.

aclSource attribute

Each entry has an associated **aclSource**. This reflects the DN that the ACL is associated. This attribute is kept and managed by the server, but might be retrieved for administrative purposes. This attribute cannot be set, only retrieved.

The derivation of **aclSource** is further explained in "Propagating ACLs" on page 455.

entryOwner attribute

Each entry has an associated **entryOwner** which are, in essence, the administrators for a particular entry. The **entryOwner** is allowed to be a user or a group, like what is allowed within an **aclEntry** attribute. If the server compatibility level is 6 or greater, a search filter is allowed to be specified to deny or grant administrator rights to users or groups. See "serverCompatLevel {3 | 4 | 5 | 6 | 7| 8}" on page 125 for more information about server compatibility.

When specifying a user or group distinguished name in an **entryOwner** attribute value, the **access-id, group**, or **role** portions of the value are optional and are accepted for compatibility with older levels of the LDAP server. If replicating to non-z/OS IBM TDS, one of these prefixes are required. The distinguished name that is used does not need to be the name of any entry in the directory. The distinguished name is the name that represents the user that has authenticated to the directory server or the group that the user is a member of.

Entry owners are not constrained by permissions given in the **aclEntry**. They have total access to any entry attribute, and can add and delete entries as they want.

Entry owners, LDAP administrators with the appropriate authority, and users who have write permission for **restricted** attributes are the only people who are allowed to change the attributes related to access control. See Chapter 9, "Administrative group and roles," on page 159 for more information about administrative role authority. If a backend is configured for basic replication as a peer or read-only replica, only an LDAP administrator and the **peerServerDN** or **masterServerDN** can set the access control attributes within the backend directory. If a subtree is configured for advanced replication, only an LDAP administrator and the **ibm-slapdMasterDN** specified on the replication agreement can set the access control attributes within the configured replication context.

The **entryOwner** attribute value is defined as a directory string.

When the **ownerFilter** scope is not specified, a search using the **entryOwner** attribute matches against the distinguished name in the value. An **entryOwner**

value in this format is normalized following the matching rules for a distinguished name. Two **entryOwner** attributes in this format are considered to be the same if they have the same distinguished name.

When the **ownerFilter** scope is specified, a search using the **entryOwner** attribute matches against the scope, filter, and the action (**grant** | **deny**) in the search filter. An **entryOwner** in this format is normalized by normalizing the scope, filter, and the action. Two **entryOwner** attributes in this format are considered to be the same if they have the same filter and action.

Note: When a filter is specified in an **aclEntry** attribute, an LDAP administrator's permissions might be reduced. However, when filters are specified in **entryOwner** attribute values, they do not reduce the effective permissions of an LDAP administrator.

Syntax

Following is the **entryOwner** attribute value syntax:

entryOwner: *entryOwner_value*[2]

subject_DN :- valid DN, represents the object that privileges are granted.

filter :- valid search filter, using the following attributes only: **ibm-filterSubject, ibm-filterIP, ibm-filterTimeOfDay, ibm-filterDayOfWeek, ibm-filterBindMechanism**, and **ibm-filterConnectionEncrypted**. See "ACL filters" on page 441 for more information.

Scope of protection

The scope of the protection is based on the following types of privilege attributes:

access-id
The distinguished name of an entry to grant administrator rights to.

group The distinguished name of the group entry to grant administrator rights to.

role The distinguished name of the group entry to grant administrator rights to.

ownerFilter
The **entryOwner** filter, that if evaluates to true, grants or denies entry owner permissions to the entry. The server compatibility must be 6 or greater to use an **ownerFilter** in an **entryOwner** attribute. See "serverCompatLevel {3 | 4 | 5 | 6 | 7| 8}" on page 125 for more information about server compatibility level.

ownerPropagate attribute

Owner propagation works 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.

---

2. where,
entryOwner_value :- [**access-id:** | **group**: | **role**:]*subject_DN*
or,
entryOwner_value :- **ownerFilter:***filter*:**grant** | **deny**

ownerSource attribute

Each entry also has an associated **ownerSource**. This reflects the DN that 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.

ACL filters

The access granted to a subject can be altered by using filters on **aclEntry** and **entryOwner** with the **aclFilter** and **ownerFilter** scope of protection. The server compatibility must be 6 or greater to use search filters in **aclEntry** and **entryOwner** attribute values. See "serverCompatLevel {3 | 4 | 5 | 6 | 7| 8}" on page 125 for more information about server compatibility level.

The syntax of specifying a filter in an **aclEntry** attribute is:

aclFilter:*filter*:*operation*[*granted_rights*]

See "aclEntry attribute" on page 434 for more information about *granted_rights*.

The syntax of specifying a filter in an **entryOwner** attribute is:

ownerFilter:*filter*:*action*

where,

filter :- An IETF RFC 2254 (RFC 2254: *The String Representation of LDAP Search Filters*) compliant LDAP search filter by using the attributes below.

operation :- **union** | **replace** | **intersect**

action :- **grant** | **deny**

A *filter* can use only the following attributes:

ibm-filterSubject
: This attribute is used to filter a distinguished name. It can be a bind DN, an alternate DN, a pseudo DN, or a group DN. The attribute can be used, for example, in a filter to reduce ACL permissions for a specific group.

ibm-filterIP
: This attribute is used to filter the IPv4 or IPv6 address of a client connection. The value can be any syntactically valid IPv4 or IPv6 address, with or without a trailing wildcard. The supported syntax for IPv6 addresses is defined in IETF RFC 2373 (RFC 2373: *IP Version 6 Addressing Architecture*). The wildcard can be specified within any IPv4 octet or in any IPv6 group, and must be the final character in the string. For example, the following are all valid:

  - 124.*
  - 124.153.242*
  - 05DC:0001:0000:0000:0000:0000:0000:2*
  - 5DC:1::2*

  Note: The final two addresses listed above are equivalent. All addresses are fully expanded. In other words, insignificant leading zeros are added to

each IPv4 octet or IPv6 group to expand it to the maximum number of digits, except for any octet or group with a wildcard is not expanded. If a wildcard is specified with the `::` IPv6 syntax, the wildcard is shifted to the final group of the address. Following are some examples:

Table 62. **ibm-filterIP** *expansion examples*

Address	Fully expanded address
1.002.03.4	001.002.003.004
01.02.03*	001.002.03*
1:2:3:4:5:6:7:8	0001:0002:0003:0004:0005:0006:0007:0008
1::8	0001:0000:0000:0000:0000:0000:0000:0008
01:2:30:4:05:6:700:08*	0001:0002:0030:0004:0005:0006:0700:08*
::FFFF:129.100.242.10	0000:0000:0000:0000:0000:FFFF:129.100.242.010
1::*	0001:0000:0000:0000:0000:0000:0000:*

ibm-filterTimeOfDay
This attribute is used to filter the time of day that the directory entry is accessed. The value is the `hh:mm` format of 24 hour time, with `hh` ranging from 00 to 23 and mm ranging from 00 to 59. This can be used, for example, to grant access only during a certain time of day.

ibm-filterDayOfWeek
This attribute is used to filter the day of week that the directory entry is accessed. The value is an integer mapping the days of the week as follows: Sunday = 0, Monday = 1, Tuesday = 2, Wednesday = 3, Thursday = 4, Friday = 5, Saturday = 6. This can be used, for example, to grant access only during certain days of the week.

ibm-filterBindMechanism
This attribute is used to filter the bind mechanism used to connect to the LDAP server. The following string values can be used to represent bind mechanisms: **SIMPLE, EXTERNAL, CRAM-MD5, DIGEST-MD5**, and **GSSAPI**. This can be used, for example, to deny access for SIMPLE binds.

ibm-filterConnectionEncrypted
This attribute is used to filter whether encryption is used to access the LDAP server. The valid values are **TRUE** and **FALSE**. This can be used, for example, to deny access for non-SSL binds or SSL binds done with no cipher specifications.

The *operation* value is required for **aclFilter** values, and specifies the way that ACL filters are applied:

replace
The effective permission is replaced by the ACL filter permission. For example, to grant clients from a given subnetwork a specific set of permissions only, use **replace**.

union The effective permission is joined with the ACL filter permission. This is used to expand permissions when granting, and reduce permissions when denying. For example, to grant clients from a given subnetwork a set of permissions, at a minimum, use **union**.

intersect
The effective permission is intersected with the ACL filter permission. This is used to reduce permissions. For example, to grant clients from a given subnetwork a set of permissions, if and only if they already have those permissions, use **intersect**.

The *action* value is required for **ownerFilter** values, and must be set to either grant, to grant entry owner access to the entry, or deny, to deny entry owner access when the LDAP search filter evaluates to true.

Filters using incorrect filter syntax, filter attributes, or operation values fail when an attempt is made to add or modify the incorrect **aclEntry** or **entryOwner** attribute value.

Note that unlike the **aclEntry** attribute, the **entryOwner** attribute cannot reduce permissions for an administrator.

Initializing ACLs with TDBM or LDBM

The TDBM or LDBM backend adds an ACL to each suffix entry if no **aclEntry** value is specified during the add of this entry (whether the add was done by using **ldapadd** or **ldif2ds**). 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 root administrator DN (**adminDN** configuration option), along with **ownerPropagate TRUE**.

Default ACLs with LDBM or 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: group:cn=anybody: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 root administrator DN (**adminDN** configuration option) 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 server is started with GDBM configured for the first time, the LDAP server creates the change log suffix entry, `cn=changelog`. The `cn=changelog` suffix entry is created with an **entryOwner** value set to the root administrator DN (**adminDN** configuration option) while the **aclEntry** attribute value is set such that non-LDAP administrators do not have access to the changelog entries. The **aclEntry** and **entryOwner** values are propagated in the GDBM backend. The ACL for the GDBM backend is:

```
aclEntry: group=cn=Anybody
```

Only the **aclEntry** and **entryOwner** attributes can be modified. LDAP administrators with the appropriate authority can also access the changelog entries.

See Chapter 9, "Administrative group and roles," on page 159 for more information about administrative role authority.

When GDBM is configured to be file-based, the **aclEntry** and **entryOwner** attributes can be entirely deleted, in which case the default ACL is used. See "Default ACLs with LDBM or TDBM" on page 443 for more information. When GDBM is configured to be DB2-based, these attributes cannot be entirely deleted. 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 if change logging is enabled (the GDBM backend is configured), even if change logging is not on. Change log entries cannot be modified to override the inherited ACL values from the change log suffix entry.

Initializing ACLs with CDBM

When the LDAP server is started with CDBM configured for the first time, the LDAP server creates the following entries:

- `cn=ibmpolicies`
- `cn=pwdpolicy,cn=ibmpolicies` (if server compatibility level is 6 or greater)
- `cn=configuration`
- `cn=Replication,cn=configuration`
- `cn=Log Management,cn=configuration`
- `cn=Replication,cn=Log Management,cn=configuration`
- `cn=admingroup,cn=configuration` (if server compatibility level is 7 or greater)
- `cn=safadmingroup,cn=configuration` (if server compatibility level is 7 or greater)

The `cn=ibmpolicies` suffix entry is created with the same initial ACL as a TDBM or LDBM suffix, that allows read access to anybody and propagates the **aclEntry** and **entryOwner** values. Therefore, only LDAP administrators with the appropriate authority can update the `cn=ibmpolicies` suffix. The **aclEntry** and **entryOwner** values can be modified. If the **aclEntry** and **entryOwner** values are deleted, the default ACL is used.

The `cn=configuration` suffix entry is created with an **entryOwner** value set to the root administrator DN (**adminDN** configuration option) while the **aclEntry** attribute value is set such that non-LDAP administrators do not have access to `cn=configuration` entries. The **aclEntry** and **entryOwner** values are propagated in the `cn=configuration` suffix of the CDBM backend. The `cn=configuration` ACL is:

```
aclEntry: group=cn=Anybody
```

Note:

1. It is suggested that you do not entirely delete the **aclEntry** and **entryOwner** values. The default ACL is used if they are deleted and allows users other than LDAP administrators with the appropriate authority access to sensitive configuration related data.
2. See Chapter 9, "Administrative group and roles," on page 159 for more information about administrative role authority.

Initializing ACLs with schema entry

When the LDAP server is started for the first time, the LDAP server creates the LDAP server schema entry, `cn=schema`. The entry is created with the same initial ACL as a TDBM or LDBM suffix that allows read access to anyone. Therefore, only an LDAP root or schema administrator can update the schema. The **aclEntry** and **entryOwner** values can be modified.

Access determination

The same bound user might be granted different access permissions to an entry if:

- the user is an LDAP root administrator and the server is, or is not, in maintenance mode
- the user is the **masterserverDN** or **peerserverDN**
- the user is the entry owner
- the user has specific access permissions set for:
  - the bind DN,
  - alternate DNs,
  - pseudo DNs,
  - groups the bind or alternate DNs are members of
- the user has matching ACL filters

See "Querying effective permissions" on page 452 for using the **GetEffectiveACL** extended operation in the **ldapexop** utility.

The z/OS LDAP server uses the following algorithm to determine the permissions to grant a bound user:

Note: When you see "additional access information", it refers to:

- the IP address of the client connection
- the bind mechanism used to bind to the LDAP server
- if encryption was used to bind to the LDAP server
- the time of day that the directory entry was accessed
- the day of week that the directory entry was accessed

An LDAP root administrator, **masterServerDN, peerServerDN**, and **entryOwner** matches are first evaluated:

1. if the bind DN is an LDAP root administrator and the server is in maintenance mode, then
   - full access is given
   - then, the algorithm ends and the server continues to process the request
2. if the bind DN is the **masterServerDN** or **peerServerDN**, then
   - full access is given
   - then, the algorithm ends and the server continues to process the request
3. if the bind DN is an LDAP administrator with the appropriate authority (see Chapter 9, "Administrative group and roles," on page 159 for more information about administrative role authority), then
   - full access is first applied
   - then, permissions for all **aclFilter** values matching the LDAP administrator's bind DN and additional access information are applied
   - finally, the algorithm ends and the server continues to process the request

4. if the bind DN matches a base **entryOwner**, or the bind DN and additional access information match an **ownerFilter**, then
   - full access is given if no matching **ownerFilter** specify the deny operator
   - then, the algorithm ends and the server continues to process the request
5. if the alternate DN matches a base **entryOwner**, or the alternate DN and additional access information match an **ownerFilter**, then
   - full access is given if no matching **ownerFilter** specify the deny operator
   - then, the algorithm ends and the server continues to process the request
6. if the group DN that the bind or alternate DNs belongs to matches a base **entryOwner**, or the group DNs and additional access information match an **ownerFilter**, then:
   - full access is given if no matching **ownerFilter** specify the deny operator
   - then, the algorithm ends and the server continues to process the request

If a match is not found or an **ownerFilter** denied access above, then **aclEntry** values are evaluated as follows:

1. if no **aclEntry** attributes are specified, then the default permissions are applied, or
2. if the bind is anonymous, and a base **aclEntry** matches the **cn=anybody** pseudo DN, or filter in an **aclFilter** component match the **cn=anybody** pseudo DN and additional access information, then:
   - See **Step a** on page step a.
3. if a base **aclEntry** value matches the bind DN or filter in an **aclFilter** component matches the bind DN and additional access information, then:
   - See **Step a** on page step a.
4. if a base **aclEntry** value matches the alternate DN or filter in an **aclFilter** component matches the alternate DN and additional access information, then:
   - See **Step b** on page step a.
5. if the entry DN matches the bind DN, and a base **aclEntry** matches the **cn=this** pseudo DN, or filter in an **aclFilter** component match the **cn=this** pseudo DN and additional access information, then:
   - See **Step a** on page step a.
6. if the entry DN matches an alternate DN, and a base **aclEntry** matches the **cn=this** pseudo DN, or filter in an **aclFilter** component match the **cn=this** pseudo DN and additional access information, then:
   - See **Step a** on page step a.
7. if a base **aclEntry** value matches the group DN that the bind or alternate DNs belongs to or filter in an **aclFilter** component match the group DN and additional access information, then:
   - See **Step b** on page step a.
8. if the bind DN is authenticated, and a base **aclEntry** matches the **cn=authenticated** pseudo DN, or filter in an **aclFilter** component match the **cn=authenticated** pseudo DN and additional access information, then:
   - See **Step a** on page step a.
9. if the bind is authenticated, and a base **aclEntry** matches the **cn=anybody** pseudo DN, or filter in an **aclFilter** component match the **cn=anybody** pseudo DN and additional access information, then:
   - See **Step a** on page step a.
10. the bound user does not receive any permissions

Step a. The algorithm applies permissions and ends as follows:

- any matching base permissions are first applied
- then, any matching **aclFilter** permissions are applied
- finally, the algorithm ends and the server continues to process the request

Step b.
- the union of matching base permissions is first applied
- then, any matching **aclFilter** permissions are applied
- finally, the algorithm ends and the server continues to process the request

Note: In each of these steps, permissions for matching filters can be set if base permissions are set or not.

If at least one **aclEntry** matches, access to system attributes are always granted read, search, and compare permissions, unless they are explicitly overridden.

The way **aclFilter** permissions are applied depends on the value specified for the operation (replace, union, and intersect). If there are one or more matching **aclFilters**, up to three temporary ACLs are formed from the union of all permissions for each operation. These are applied to the bound user's effective permission as follows:

1. If there are filter permissions for the replace operation, the bound user's base effective ACL is replaced by the replace filter permissions to form a new effective permission.
2. If there are any filter permissions for the union operation, the bound user's new effective permission becomes the union of the union filter permissions and the existing effective permission. The existing effective permission is produced by step 1, or if step 1 did not apply, it is the base effective permission.
3. If there are any filter permissions for the intersect operation, the bound user's new effective permission becomes the intersect of the intersect filter permissions and the existing effective permission. The existing effective permission is produced by step 2, or if step 2 did not apply, then it is the existing effective permission produced by step 1, or if step 1 did not apply, it is the base effective permission.

When using attribute-level permissions or 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. 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 priority over a grant permission (for the same item specified) while attribute-level permissions take precedence over access-class permissions.

A similar grant or deny precedence exists for the grant or deny support provided for **ownerFilter**. A deny always takes precedence over a grant of **entryOwner** access.

Access determination examples

The following are examples that illustrate the permissions that users have for entries and attribute types.

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, are 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.

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 is only granted read, write, search, and compare to **normal** attributes. A member of both group1 and group2 is 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` also has 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` still has write access to the **cn** attribute by virtue of **cn** having an access class of **normal**.

Example 9:

```
aclEntry: cn=Ken, o=Your Company:normal:rsc
aclEntry: aclFilter: (&(|(ibm-filterSubject=cn=Ken, o=Your Company)
 (ibm-filterIP=129.176.132.*))(ibm-filterTimeOfDay>=09:00)
 (ibm-filterTimeOfDay<=17:00)(ibm-filterDayOfWeek>=1)
 (ibm-filterDayOfWeek<=5)):union:normal:w
```

In this example, `cn=Ken, o=Your Company` is granted read, write, search, and compare access on **normal** attributes when authenticated from any IP address on Monday through Friday from 9:00 a.m. to 5:00 p.m.. On any other time or day, `cn=Ken, o=Your Company` is only granted read, search, and compare access on **normal** attributes.

If another user binds from IP address 129.176.132.28 on Monday through Friday from 9:00 a.m. to 5:00 p.m., they are granted write access on **normal** attributes.

Example 10:

```
aclEntry: group:cn=group1, o=Your Company:system:rsc:normal:sw
aclEntry: aclFilter: (&(ibm-filterSubject=cn=group1, o=Your Company)
 (ibm-filterIP=129.176.*)(ibm-filterBindMechanism=CRAM-MD5)
 (ibm-filterConnectionEncrypted=true)):intersect:system:rsc:normal:s
```

In this example, if `cn=Ken,o=Your Company` is a member of `cn=group1,o=Your Company`, binds from IP address 129.176.113.76 using a CRAM-MD5 mechanism, over an encrypted SSL connection. `cn=Ken,o=Your Company` is granted `system:rsc` and `normal:s` permissions. Note the intersect operation restricted permissions by disposing of `normal:w` permissions.

Example 11:

```
aclEntry: access-id:cn=Joe,dc=yourcompany,dc=com,o=IBM:normal:r
aclEntry: group:cn=group1,dc=yourcompany,dc=com:normal:rw
aclEntry: group:cn=group2,dc=yourcompany,dc=com:critical:rw
aclEntry: aclFilter:(&(ibm-filterSubject=cn=group1,dc=yourcompany,dc=com)
 (ibm-filterIP=129.176.*)):union:normal:sc
```

In this example, if cn=Joe,dc=yourcompany,dc=com,o=IBM is a member of cn=group1,dc=yourcompany,dc=com and binds from IP address 129.176.53.92, cn=Joe,dc=yourcompany,dc=com,o=IBM is only granted read permission on **normal** attributes, for the following reasons:

- There is a specific access-id value for cn=Joe,dc=yourcompany,dc=com,o=IBM, therefore, only that base access-id ACL is granted, and no group ACLs are granted. cn=Joe,dc=yourcompany,dc=com,o=IBM's effective permission, before any filters are applied, is normal:r.
- The resulting subject that is tested for membership in the filter is cn=Joe,dc=yourcompany,dc=com,o=IBM. It does not match, therefore, the ACL filter does not apply.

A member of cn=group1,dc=yourcompany,dc=com that binds from IP address 172.191.214.98 is only granted read and write permissions on **normal** attributes.

A member of both cn=group1,dc=yourcompany,dc=com and cn=group2,dc=yourcompany,dc=com that binds from IP address 129.176.98.112, is granted read, write, search, and compare access on **normal** attributes (because of the union with the filter), and is granted read and write access on **critical** attributes.

Example 12:

```
aclEntry: access-id:cn=Mary,dc=yourcompany,dc=com,o=IBM:normal:rw
aclEntry: group:cn=group1,dc=yourcompany,dc=com:normal:rwsc
aclEntry: aclFilter:(|(ibm-filterSubject=cn=group1,dc=yourcompany,dc=com)
 (ibm-filterIP=129.176.92.*)):intersect:normal:rsc:critical:r
```

In this example, if cn=Mary,dc=yourcompany,dc=com,o=IBM (who is not a member of group1,dc=yourcompany,dc=com) binds from IP 129.176.92.113, cn=Mary,dc=yourcompany,dc=com,o=IBM is granted read permission on **normal** attributes. This is because cn=Mary,dc=yourcompany,dc=com,o=IBM's effective permission is determined by the intersect of normal:rw and the ACL filter. Note the intersect operation restricted permissions by disposing of normal:wsc and critical:r while keeping normal:r permissions.

Example 13:

```
aclEntry: group:cn=group1,dc=yourcompany,dc=com:normal:rwsc
aclEntry: aclFilter:(ibm-filterIP=129.176.*):replace:normal:rw
aclEntry: aclFilter:(!(ibm-filterSubject=cn=group1,dc=yourcompany,dc=com))
 :replace:normal:deny:w
```

In this example, a member of cn=group1,dc=yourcompany,dc=com that binds from IP address 129.176.29.52 is granted read and write access to **normal** attributes. The base **aclEntry** for cn=group1,dc=yourcompany,dc=com is first applied, and is then replaced by the **aclEntry** with **aclFilter** for ibm-filterIP=129.176.*.

Example 14:

```
aclEntry: aclFilter:(ibm-filterIP=129.176.29.52):intersect:normal:sc
aclEntry: aclFilter:(ibm-filterIP=129.176.*):replace:normal:r
aclEntry: aclFilter:(ibm-filterSubject=cn=group1, o=Your Company)
 :union:critical:r
aclEntry: aclFilter:(ibm-filterIP=129.176.29*):union:sensitive:r
aclEntry: aclFilter:(ibm-filterSubject=cn=Mary, o=Your Company)
 :intersect:normal:r
aclEntry: group:cn=group1, o=Your Company:normal:rw
aclEntry: aclFilter:(ibm-filterSubject=cn=group1, o=Your Company)
 :replace:normal:rwsc
```

In this example, if cn=Mary, o=Your Company, a member of cn=group1, o=Your Company, binds from IP address 129.176.29.52, cn=Mary, o=Your Company's effective permission is determined by all of the **aclEntry** values above, as follows:

1. cn=Mary, o=Your Company's base effective permission is granted read and write permissions on **normal** attributes. This is because cn=Mary, o=Your Company is a member of cn=group1, o=Your Company.
2. The replace filters that match for cn=Mary, o=Your Company are then joined to produce one replace filter with read, write, search, and compare access on **normal** attributes. These permissions now become cn=Mary, o=Your Company's effective permission, replacing Mary base effective permission.
3. The union filters that match for cn=Mary, o=Your Company are then joined to produce one union ACL with read permissions on **sensitive** and **critical** attributes. These permissions are joined with cn=Mary, o=Your Company's effective permission that was produced in step 2. This produces cn=Mary, o=Your Company's new effective permission, granting read, write, search, and compare permissions on **normal** attributes, and read permissions on **sensitive** and **critical** attributes.
4. Finally, the intersect filters that match for cn=Mary, o=Your Company are joined to produce one intersect ACL with read, search, and compare access on **normal** attributes. These permissions are intersected with the effective permission produced by step 3, to produce cn=Mary, o=Your Company's final effective permission, giving Mary read, search, and compare access on **normal** attributes.

Example 15:

```
entryOwner: ownerFilter:(&(ibm-filterSubject=cn=Ken, o=Your Company)
 (ibm-filterIP=129.176.132.*))
```

In this example, cn=Ken, o=Your Company is given owner access only when bound from an IP address within the 129.176.132.* subnetwork.

Example 16:

```
entryOwner: access-id:cn=Ken, o=Your Company
entryOwner: ownerFilter:(&(ibm-filterSubject=cn=Ken, o=Your Company)
 (ibm-filterIP=129.176.132.*)):deny
```

In this example, cn=Ken, o=Your Company is granted owner access when bound from an IP address that is not within the 129.176.132.* subnetwork. However, cn=Ken, o=Your Company is denied owner access when bound from an IP address that is within the 129.176.132.* subnetwork.

Search

In order to read an attribute from the directory, the user must have read permission for the specific attribute or for the attribute access class that the attribute belongs.

Filter

In order to use an attribute in a search filter supplied on a search operation, the user must have search permission for the specific attribute or for the attribute access class that the attribute belongs.

Compare

In order to perform a compare operation on an attribute/value combination, the user must have compare permission for the specific attribute or for the attribute class that the attribute belongs.

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, allow the **aclEntry** be

```
group:cn=Anybody:normal:rsc:sensitive:c:critical:c
```

and allow the client to 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. Users performing anonymous searches are given the permission granted to the **cn=Anybody** group. In this example, permission exists to the filter because **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 because **cn=Anybody** does not have read permissions on the **sensitive** and **critical** attribute access classes.

Querying effective permissions

When filters are specified in **aclEntry** or **entryOwner** attribute values in the directory, it might be difficult to determine the permissions that users or groups have to entries. To ease in the determination of effective ACLs, the **ldapexop** utility provides the **GetEffectiveACL** extended operation. See “ldapexop utility” on page 255 for more information.

This extended operation in the **ldapexop** utility allows the specification of search criteria and bound user's information (such as the bind distinguished name, time of day, the day of the week, and IP address where the user is authenticating from). By specifying the search criteria and bound user's information, an LDAP root administrator is allowed to simulate the effective ACLs for multiple users in the directory.

This extended operation returns the following information for each requested entry:

- the entry DN to which access was requested
- the subject and all of its alternate DNs and group DNs for which access was calculated for
- the source attribute values (**aclEntry, aclPropagate, aclSource, entryOwner, ownerPropagate**, and **ownerSource**) in effect for the entry
- the applicable attribute values (**aclEntry** and **entryOwner**) used to form the effective permissions
- the calculated effective access class permissions
- the calculated effective attribute permissions

This example performs the **GetEffectiveACL** extended operation for each entry returned on the subtree search of the `dc=yourcompany,dc=com` subtree. The requested subtree search uses `dc=yourcompany,dc=com` as the baseDN, with a filter of `"objectclass=*"`, a search size limit of `100`, a search time limit of `10` seconds, and no alias dereferencing. Based on these returned search entries, the **GetEffectiveACL** extended operation calculates the effective ACLs for user `cn=Joe`

Shmoe,ou=users,dc=yourcompany,dc=com when a simple bind is done from IP address 129.176.132.92 at 18:30 on a Saturday over a secure SSL connection.

```
ldapexop -D adminDn -w adminPw -op geteffectiveacl -filter "objectclass=*"
 -base "dc=yourcompany,dc=com" -s sub -a never -z 100 -l 10
 -dn "cn=Joe Shmoe,ou=users,dc=yourcompany,dc=com" -ip 129.176.132.92
 -time 18:30 -day 6 -mech SIMPLE –encrypt
```

```
#ENTRY INFORMATION:
dn: dc=yourcompany,dc=com
#SUBJECT INFORMATION:
#Bind DN:
dn: cn=Joe Shmoe,ou=users,dc=yourcompany,dc=com

#Alternate DNs:
dn: cn=alt_01
dn: cn=alt_02

#Group DNs:
dn: cn=group_01
dn: cn=group_02

#SOURCE ATTRIBUTE VALUES:
aclEntry: group:cn=Anybody:normal:rsc:system:rsc
aclPropagate: TRUE
aclSource: dc=yourcompany,dc=com
entryOwner: cn=Admin
ownerPropagate: TRUE
ownerSource:dc=yourcompany,dc=com

#APPLICABLE ATTRIBUTE VALUES:
aclEntry: group:cn=Anybody:normal:rsc:system:rsc

#EFFECTIVE ACCESS-CLASS PERMISSIONS:
normal: grant:rsc
system: grant:rsc


#ENTRY INFORMATION:
dn: ou=users,dc=yourcompany,dc=com

#SUBJECT INFORMATION:
#Bind DN:
dn: cn=Joe Shmoe,ou=users,dc=yourcompany,dc=com

#Alternate DNs:
dn: cn=alt_01
dn: cn=alt_02

#SOURCE ATTRIBUTE VALUES:
aclEntry: group:cn=Anybody:normal:rsc:system:rsc
aclPropagate: TRUE
aclSource: dc=yourcompany,dc=com
entryOwner: cn=Joe Shmoe,ou=users,dc=yourcompany,dc=com
ownerPropagate: TRUE
ownerSource: dc=yourcompany,dc=com

#APPLICABLE ATTRIBUTE VALUES:
entryOwner: cn=Joe Shmoe,ou=users,dc=yourcompany,dc=com

#EFFECTIVE ACCESS-CLASS PERMISSIONS:
restricted:grant:rwsc
system:grant:rwsc
critical:grant:rwsc
sensitive:grant:rwsc
normal:grant:rwsc
object:grant:ad


#ENTRY INFORMATION:
dn: cn=Joe Shmoe,ou=users,dc=yourcompany,dc=com
```

```
#SUBJECT INFORMATION:
#Bind DN:
dn: cn=Joe Shmoe,ou=users,dc=yourcompany,dc=com

#SOURCE ATTRIBUTE VALUES:
aclEntry: access-id:cn=Joe Shmoe,ou=users,dc=yourcompany,dc=com:normal:rsc:
 sensitive:rsc:critical:rsc
aclEntry: aclFilter:(&(ibm-filterSubject=cn=Joe Shmoe,ou=users,dc=yourcompan
 y,dc=com)(ibm-filterIP=129.176.132.*)(|(ibm-filterTimeOfDay<09:00)(ibm-filt
 eRTimeOfDay>17:00))(|(ibm-filterDayOfWeek<1)(ibm-filterDayOfWeek>5))):union
 :object:ad:normal:w
aclEntry: group:cn=Anybody:normal:rsc
aclPropagate: TRUE
aclSource: cn=Joe Shmoe,ou=users,dc=yourcompany,dc=com
entryOwner: cn=Admin
ownerPropagate: TRUE
ownerSource: dc=yourcompany,dc=com

#APPLICABLE ATTRIBUTE VALUES:
aclEntry: access-id:cn=Joe Shmoe,ou=users,dc=yourcompany,dc=com:normal:rsc:
 sensitive:rsc:critical:rsc
aclEntry: aclFilter:(&(ibm-filterSubject=cn=Joe Shmoe,ou=users,dc=yourcomany,
 dc=com)(ibm-filterIP=129.176.132.*)(|(ibm-filterTimeOfDay<09:00)
 (ibm-filterTimeOfDay>17:00))(|(ibm-filterDayOfWeek<1)
 (ibm-filterDayOfWeek>5))):union:object:ad:normal:w

#EFFECTIVE ACCESS-CLASS PERMISSIONS:
normal: grant:rwsc
sensitive: grant:rsc
critical: grant:rsc
object: grant:ad

#ENTRY INFORMATION:
dn: cn=Corey,ou=users,dc=yourcompany,dc=com

#SUBJECT INFORMATION:

#Bind DN:
dn: cn=Joe Shmoe,ou=users,dc=yourcompany,dc=com

#SOURCE ATTRIBUTE VALUES:
aclEntry: access-id:cn=Corey,ou=users,dc=yourcompany,dc=com:normal:rsc:
 sensitive:rsc:critical:rsc
aclEntry: AclFilter: (&(ibm-filterSubject=cn=Corey,ou=users,
 dc=yourcompany,dc=com)(ibm-filterIP=129.176.132.*)(|
 (ibm-filterTimeOfDay<09:00)(ibm-filterTimeOfDay>17:00))(|
 (ibm-filterDayOfWeek<1)(ibm-filterDayOfWeek>5))):union:object:ad:normal:w
aclEntry: group:cn=Anybody:normal:rsc
aclEntry: access-id:cn=Joe Shmoe,ou=users,dc=yourcompany,dc=com:normal:
 rsc:at.telephoneNumber:deny:rsc:at.cn:deny:rsc
aclPropagate: TRUE
aclSource: cn=Corey,ou=users,dc=yourcompany,dc=com
entryOwner: cn=Admin
ownerPropagate: TRUE
ownerSource: dc=yourcompany,dc=com

#APPLICABLE ATTRIBUTE VALUES:
aclEntry: access-id:cn=Joe Shmoe,ou=users,dc=yourcompany,dc=com:normal:
 rsc:at.telephoneNumber:deny:rsc:at.cn:deny:rsc

#EFFECTIVE ACCESS-CLASS PERMISSIONS:
normal: grant:rsc

#EFFECTIVE ATTRIBUTE PERMISSIONS:
at.cn: deny:rsc
at.telephoneNumber: deny:rsc
```

Propagating ACLs

ACLs can be set on any entry in the hierarchy. ACLs (including ACL filters) can propagate through the directory hierarchy. These ACLs, called propagating ACLs, have the **aclPropagate** attribute set to **TRUE**. All descendants of this entry 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 propagates down to all descendants. If set to **FALSE**, the ACL is not propagated; it instead becomes an override ACL. The ACL is not propagated 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. If there is no propagating ACL, the entry receives the default 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 through the tree until another explicitly-set propagating ACL is found.

The same rules apply to propagating the entry owner permissions (including ACL filters for entry owners) based on the **ownerPropagate** attribute.

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 is set on `ou=deptXYZ, o=IBM, c=US`. No ACL is set on the child `cn=personA, ou=deptXYZ, o=IBM, c=US`. Therefore, the child 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 LDAP 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 is set on `o=IBM, c=US`. An override ACL is set (**aclPropagate** is **FALSE**) on the child **ou=deptXYZ, o=IBM, c=US**. Therefore, the ACL set at **ou=deptXYZ, o=IBM, c=US** pertains only to that particular entry.

The child `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 root administrator DN or **adminDN** configuration option is `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 attributes 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 about attributes in the **critical** attribute access class. Unauthenticated, and 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 because it is the owner of the entry. As always, an LDAP root or directory data administrator have unrestricted access to the entry.

Access control groups

Access control groups provide a mechanism for applying the same **aclEntry** or **entryOwner** attribute values to an entry for multiple users without having to create an explicit **aclEntry** or **entryOwner** for each user.

For the LDBM, CDBM, and TDBM backends, the following object classes are evaluated as access control group entries: **accessGroup, accessRole, groupOfNames, groupOfUniqueNames, ibm-staticGroup, groupOfUrls, ibm-dynamicGroup,** and **ibm-nestedGroup**. See Chapter 23, "Static, dynamic, and nested groups," on page 419 for more information about static, dynamic, and nested groups.

Associating DNs, access groups, and additional bind and directory entry access information with a bound user

After a successful bind request, a bind distinguished name is associated with the bound user. Depending upon the bind method, there can also be a list of alternate DNs associated with the bound user.

- For a simple bind, the bind DN is the DN specified in the bind request. There must be an entry in LDAP with that DN. The entry can be in an LDBM, SDBM, CDBM, or TDBM backend, or in a client operation plug-in extension. There are no alternate DNs.
- For a CRAM-MD5 bind, the bind request must specify a DN or a user name. If a DN is specified, there must be an entry in LDAP with that DN. If a user name is specified, there must be an entry in LDAP that contains the user name as a **uid** attribute value. If both a DN and a user name are specified, there must be an entry in LDAP with that DN and the user name must be a **uid** attribute value in that entry. In all of these cases, the bind DN is the DN of the entry. The entry can be in an LDBM, CDBM, or TDBM backend, or in a client operation plug-in extension. There are no alternate DNs. See Chapter 21, "CRAM-MD5 and DIGEST-MD5 authentication," on page 413 for more information.
- For a DIGEST-MD5 bind, the bind request must specify a user name and may optionally contain an authorization DN. If only a user name is specified, there must be an entry in LDAP that contains the user name as a **uid** attribute value. If both a user name and an authorization DN are specified, there must be an entry in LDAP with the authorization DN as its DN and the user name must be a **uid** attribute value in that entry. In both cases, the bind DN is the DN of the entry. The entry can be in an LDBM, CDBM, or TDBM backend or in a client operation plug-in extension. There are no alternate DNs. See Chapter 21, "CRAM-MD5 and DIGEST-MD5 authentication," on page 413 for more information.
- For a Kerberos bind, the bind DN is `ibm-kn=`*`principal@REALM`* where *`principal@REALM`* is the Kerberos identity specified in the bind request. There cannot be an entry in an LDBM, CDBM, or TDBM backend or in a client operation plug-in extension corresponding to this DN. Each LDBM, TDBM, CDBM, and SDBM backend that **krbIdentityMap on** is specified in the LDAP server configuration file is contacted to map the Kerberos identity to alternate DNs in that backend. The client operation plug-in extensions are also contacted if they registered a **SLAPI_TYPE_ALT_NAMES** callback type routine and can contribute alternate DNs. See Chapter 19, "Kerberos authentication," on page 393 for more information.
- For a certificate (EXTERNAL) bind, the bind DN is usually the subject DN from the certificate specified in the bind request. There cannot be an entry in an

LDBM, CDBM, or TDBM backend or in a client operation plug-in extension corresponding to this DN. If there is a RACF user associated with the certificate and **replace** is specified for the **sslMapCertificate** option in the LDAP server configuration file, then an SDBM-style DN based on the RACF user name replaces the subject DN as the bind DN. If **add** is specified for the **sslMapCertificate** option, then the bind DN is not changed and the SDBM-style DN based on the RACF user name is added as an alternate DN. See "Support of certificate bind" on page 76 for more information.

After the bind DN and alternate DNs (if any) associated with the bound user are determined, the DNs of the groups the bound user belongs to are added to the bind information. If LDAP password policy is active, groups are determined during authentication time. If LDAP password policy is not active, the groups are determined at the beginning of the next non-bind request. The bind DN and group information are used in access control in LDAP operations from the bound user.

Note: Group gathering is not performed if any of the following is true:

1. The user binds as the **adminDN**, **peerServerDN**, or **masterServerDN**.
2. The **authenticateOnly** server control is specified as part of the bind request.

The groups are gathered in the following manner:

- The backend or client operation plug-in extension that contains the bind DN is contacted to contribute DNs of any group entries that contain the bind DN or any of the alternate DNs. If the bind DN is not in a backend or a client operation plug-in extension (for example, after a Kerberos bind), this step is skipped.
- Each LDBM, CDBM, or TDBM backend that has **extendedGroupSearching on** specified in the LDAP server configuration file is also contacted to contribute the DNs of any group entries in the backend that contain the bind DN or any of the alternate DNs. The client operation plug-in extensions are also contacted to contribute group DNs if they registered a **SLAPI_TYPE_GROUPS** callback type routine. Note that SDBM does not support extended group searching.

Besides bind DN, alternate DNs, and groups, additional data are added to the bind information and to the directory entry access information to further distinguish the identity of the bound user. This information is used when matching ACL filters to determine access control for LDAP operations from the bound user. Search size and time limits are also gathered using group search limits from groups to which the bound user belongs. The following additional data are added to the bound user's bind information:

- IP address of the bound user's client connection
- Bind mechanism used to bind to the LDAP server
- Whether a secure, encrypted SSL connection was used to bind to the LDAP server
- Largest size limit and time limit specified in group search limits

The following data is added to the bound user's directory entry access information when accessing an entry:

- Time of day the bound user accessed the directory entry
- Day of week the bound user accessed the directory entry

Deleting a user or a group

Deleting a user or a group does not have any cascade effect on any **aclEntry** and **entryOwner** values that include that user or group. The user or group DN is not removed from the ACLs. If another user or group is later created with the same DN, that user or group is granted the privileges of the former user or group.

Retrieving ACL information from the server

To retrieve all of the ACL information in a namespace, use the **ldapsearch** utility, 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 that 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 RFC 2251: *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.

Note:

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.
2. If the tree is larger than the **sizeLimit** configuration option in the LDAP server configuration file or on the search request or in the requester's group search limits, then some entries might not be returned. See "sizeLimit num-limit " on page 128 for more information about the **sizeLimit** configuration option.

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** utility, where *searchbase* is the starting point for the search.

Creating and managing access controls

To create and update ACLs in LDBM, TDBM, CDBM, GDBM, or the schema entry, use a tool implementing **ldap_modify** APIs, such as **ldapmodify**. The **ldapmodify** utility allows creation, modification, and deletion of any set of attributes that are associated with an entry in the directory. Because 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 LDBM, TDBM, CDBM, GDBM, or the schema entry.

See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for details on using the 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.

Because the **ldapmodify** utility is 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** utility to add ACL information.

Figure 30 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 30. Example of adding propagating ACL to existing entry in directory

The ACL added in Figure 30 is created as a propagating ACL because the **aclPropagate** attribute is not specified and is assumed to be **TRUE**. This means that the ACL applies 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.

It is not required that an administrator updates all ACL information, the examples in this section all use a root administrator when updating ACLs. Further, the use of `-h 127.0.0.1` implies that the **ldapmodify** utility is performed from the same

system that the LDAP server is running and that the LDAP server is listening on TCP/IP port 389. See the **ldapmodify** utility description in *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more details on the **-h**, **-p**, **-D**, and **-w** command-line options. The ACL attributes can be updated from any LDAP client if the user performing the updates has the correct authorization to update the ACL information.

The ACL attributes are defined to be in a special access class called **restricted**. Therefore, in order to allow someone other than an LDAP administrator to update the ACL attributes, they must either be the entry owner or have the correct authorization to **restricted** attributes. Figure 31 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
-
add: aclPropagate
aclPropagate: TRUE
-
```

Figure 31. Example of adding propagating ACL to existing entry in the directory.

The ACL added in Figure 31 allows `cn=jeanne, o=Your Company` to update the ACL information for this entry. In addition, because 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, because 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 (an administrator, for example) must be used in order to reset/change the ACL information.

Figure 32 on page 462 shows an example of adding a non-propagating ACL. A non-propagating ACL applies only to the entry that 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 32. Example of setting up a non-propagating ACL

Setting up a non-propagating ACL is like 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 might need to be updated. To do this, the **ldapmodify** utility is used. The examples in this section use the **ldapmodify** utility, however, any LDAP client application issuing LDAP modify operations to the LDAP server might be used. Therefore, modifications to ACL information do not need to be performed from the same system that 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 that exists in the ACL to disallow another person or group access to the entry that they had before.

Figure 33, Figure 34 on page 463, and Figure 35 on page 463 show examples of these modifications, in that order.

"Access determination" on page 445 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 33. Example of adding an aclEntry attribute value

In Figure 33, `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 34 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 34. Example of modifying aclPropagate attribute

In Figure 34, 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 no longer applies 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` is 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 35 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 35. Example of removing a single aclEntry attribute value

In Figure 35, 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**. Only the distinguished name part of the **aclEntry** value must be specified when deleting the value.

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 change log suffix entry, `cn=changelog`, when GDBM is DB2-based.

Figure 36 on page 464 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 36. Example of deleting an ACL from an entry

In Figure 36, the existing ACL against `cn=jeff, cn=tim, o=Your Company` is removed. This means that the ACL no longer applies to the entry. Instead, the first propagating ACL that is found in an entry above `cn=jeff, cn=tim, o=Your Company` is 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.

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, including read, write, search, and compare for all attribute classes. Entry owners can add and modify ACL information about the entries that they are specified as the owner.

Entry owners are listed in the **entryOwner** attribute. 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 that 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.

Because the **ldapmodify** utility 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** utility to add entry owner information.

Figure 37 on page 465 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 37. Example of adding a propagating set of entry owners to existing entry in the directory

The entry owners added in Figure 37 are created as a propagating set of entry owners **since** the **ownerPropagate** attribute is not specified and, therefore, assumed to be **TRUE**. This means that the entry owners apply to all entries below `cn=tim, o=Your Company` that do not already have an entry owner associated with them.

It is not required that an LDAP administrator update all entry owner information, the examples in this section all use the administrator as the entry owner updating ACLs. Further, the use of `-h 127.0.0.1` implies that the **ldapmodify** utility is performed from the same system that the LDAP server is running and that the LDAP server is listening on TCP/IP port 389. See the **ldapmodify** utility description in *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more details on the **-h**, **-p**, **-D**, and **-w** command-line options. The entry owner attributes can be updated from any LDAP client if the user performing the update has the correct authorization to update the entry owner information.

The entry owner attributes, such as the ACL attributes, are defined to be in a special access class called **restricted**. Therefore, in order to allow someone other than an LDAP administrator to update the entry owner attributes, they must either be the entry owner or have the correct authorization to **restricted** attributes. See Figure 31 on page 461 for an example of allowing users other than an LDAP administrator the ability to update entry owner information.

Figure 38 shows an example of adding a non-propagating entry owner. A non-propagating entry owner applies only to the entry that 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 38. 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 might need to be updated. To do this, the **ldapmodify** utility is used. The examples in this section use the **ldapmodify** utility, however, any LDAP client application issuing LDAP modify operations to the LDAP server might be used. Therefore, modifications to entry owner information do not need to be performed from the same system that 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 that exists in the entry owner set to disallow another person or group access to control the entry that they had control over before.

Figure 39, Figure 40, and Figure 41 on page 467 show examples of these modifications, in that order.

Figure 39 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 39. Example of adding an entryOwner attribute value

In Figure 39, `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 40 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 40. Example of modifying the ownerPropagate attribute

In Figure 40, 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 no longer applies 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 is 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 41 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 41. Example of removing a single entryOwner Attribute value

In Figure 41, 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**. Only the distinguished name part of the **entryOwner** value must be specified when deleting the value.

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** utility to delete the entire attribute (all values) from the entry.

Note: This is not allowed for the change log suffix entry, cn=changelog, when GDBM is DB2-based.

Figure 42 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 42. Example of deleting an entry owner set from an entry

In Figure 42, the existing entry owner set against cn=jeff, cn=tim, o=Your Company is removed. This means that the entry owner information no longer applies to the entry. Instead, the first propagating entry owner set that is found in an entry above cn=jeff, cn=tim, o=Your Company is 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.

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, which is 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 23, “Static, dynamic, and nested groups,” on page 419 for more information about creating, modifying, and deleting static, dynamic, and nested group entries.

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 can be used in determining whether 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 43 shows how to be added to the **aclEntry** information in an existing access control specification for an entry. Figure 44 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 43. 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 44. Example of adding a group to entry owner information

Chapter 25. Basic replication

Once the z/OS LDAP server is installed and configured, users can access the directory, add entries, delete entries, or perform search operations to retrieve particular sets of information.

Replication is a process which keeps multiple directories in sync. Through replication, a change made to one directory is propagated to one or more additional directories. In effect, a change to one directory shows up on multiple different directories.

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 basic 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 basic replication support for peer to peer replication is provided for failover support purposes. With basic peer to peer replication, there is no support for resolving 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 basic 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 directory are made to the master server. The master server is then responsible for propagating the changes to all other directories. It is important to note that while there can be multiple directories representing the same information, only one of those directories can be the master.

  Read-only replica
  Each of the additional servers which contain a directory replica. These replica directories are identical to the master directory. These servers are read-only to all users and only accept updates from their master server.

If you need more advanced replication choices, see Chapter 26, “Advanced replication,” on page 487.

Note: Basic and advanced replication are not allowed in the same server. If both are enabled in the server, the server fails to start.

A basic 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.

Basic replication is supported when the servers involved are running in single-server or in multi-server mode. See "Determining operational mode" on page 146 for more information about server operating modes.

In z/OS LDAP, basic replication is supported in the LDBM and TDBM backends but is not supported in the SDBM or GDBM backends or the schema entry.

Basic replication in a sysplex

A set of LDAP servers sharing a backend directory in a sysplex can act as a master, read-only, or peer server to LDAP servers that are not in the sysplex. Each LDAP server in the sysplex must have the same replication options (**masterServer, masterServerDN, masterServerPW, peerServerDN**, and **peerServerPW**) in the backend section of the LDAP server configuration file. Do not make an LDAP server in the sysplex a replica of another LDAP server in the sysplex for a backend directory that they are sharing.

When the set of LDAP servers in the sysplex is set up to be a master or a peer replica server and changes occur to the shared LDAP directory, the LDAP server acting as the owner of the sysplex group replicates the directory changes to the LDAP server replicas that are not in the sysplex. These replicas are identified by replica entries in the shared directory.

When the set of LDAP servers in the sysplex is set up to be a read-only or a peer replica server and directory changes occur to a master or peer LDAP server that is not in the sysplex, that LDAP server replicates the changes to an LDAP server in the sysplex, identified in a replica entry in the directory. The changes are made to the backend directory in the sysplex and are seen by all the LDAP servers sharing the directory in the sysplex.

ibm-entryuuid replication

Basic replication of the **ibm-entryuuid** attribute is performed to any LDAP server that has 1.3.18.0.2.32.3 (the OID for the entry UUID capability) as a value in the **ibm-enabledCapabilities** attribute in the root DSE. z/OS LDAP servers have this capability. If the root DSE of a replica server does not contain the required capability, then the **ibm-entryuuid** attribute are not replicated to that server, however, the entry and other attributes are not replicated.

Complex modify DN replication

Basic replication of Modify DN new superior operations are performed to any LDAP server that has 1.3.18.0.2.32.33 (the OID for the subtree move capability) or 1.3.18.0.2.32.34 (the OID for the subtree rename capability) as a value in the **ibm-enabledCapabilities** attribute in the root DSE. z/OS LDAP servers have this capability. If a replica server is not at a supported level, Modify DN new superior operations fail until the replica is removed from the replica collection.

Basic replication and ldif2ds

ldif2ds does not replicate changes when adding entries to the replicating server. If you are using **ldif2ds** 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.

In order to maintain directory integrity, a load utility (**ldif2ds**) should be used on a read-only or peer replica only when initially populating the replica directory. If a load utility is used to add entries to a replica server after initial population, these changes are not reflected in the master directory. The replica directory might give erroneous information.

Data encryption or hashing and basic replication

When encryption or hashing is configured in an LDBM or TDBM backend participating in a basic replication environment, attribute values subject to encryption or hashing based on the **pwEncryption** or **secretEncryption** configuration options are either replicated in the clear or hashed.

When configuring basic replication in an LDBM backend, the following should be considered when setting the **pwEncryption** and **secretEncryption** configuration options:

1. If the **pwEncryption** or **secretEncryption** configuration option is set to AES or DES, the attribute values eligible for encryption on add or modify requests are sent from the master or peer server to the replica or other peer server in the clear. Because these sensitive attribute values are replicated in the clear, a secure or SSL connection should be configured between the servers to protect this data while it is in transit. See "Replicating server" on page 472and "Configuring the replica" on page 477 for more information.
2. If the **pwEncryption** configuration option is set to **crypt** and replication is configured between a z/OS LDAP server and a non-z/OS LDAP server, specify **pwCryptCompat off** in the LDBM backend section of the configuration file. This setting indicates that the LDAP server should use the UTF-8 version of the crypt algorithm to hash eligible values. When eligible attribute values (for example, **userPassword**) for hashing in **crypt** are replicated between z/OS and non-z/OS LDAP servers, the password is the same on both platforms and, therefore, is usable.

   If basic replication is configured between two z/OS LDAP servers, verify the **pwCryptCompat** configuration option has the same settings on both servers. This ensures that the values are usable on both servers.
3. If the **pwEncryption** configuration option is set to any other one-way hashing method (for example, SHA, MD5, SSHA, SHA-2 or Salted SHA-2), the master or peer server replicates the tagged hashed value to the replica or other peer server. Therefore, the replica or the other peer server must support the same hashing method to ensure that the values are usable on the other server.

When basic replication is configured in a TDBM backend, the attribute values eligible for encryption or hashing on add or modify requests are always sent from the master or peer server to the replica or other peer server in the clear. A secure or SSL connection should be configured between the servers to protect this data while it is transit. See "Replicating server" on page 472 and "Configuring the replica" on page 477 for more information.

Replicating server

In order for the basic 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 and peer server must be aware of the replicating servers for the directory that it serves. See "LDAP update operations on read-only replicas" on page 478 for more information.

The replicating server becomes aware of the existence of the replica servers when entries with an object class of **replicaObject** are added to the directory. Each of these entries represents a particular replica server. The attribute/value pairs within the replica entry provide the information the replicating server needs in order to locate the replica server and send any updates to that server.

Replica entries

The **replicaObject** object class is provided in the initial schema. 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**. Values in a replica entry are recognized at server startup and when a replica entry is added or modified. The internal number of how many replication operations have been set aside (the set aside count) for a replica is not reset when the replica entry is modified. To reset the count, either the server must be restarted or the replica entry must be removed and added. See "Basic replication error log" on page 483 for more information about the set aside count.

Table 63. Replica entry schema definition (mandatory attributes)

Attribute	Description and example

replicaHost	This can be an IPv4 address, IPv6 address, or a host name of the system where the replica server is running.
	
	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 LDAP server 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**. The **replicaCredentials** attribute value is encrypted if the **secretEncryption** option is specified in the LDAP server configuration file. This improves directory security because the bind password is no longer stored in the directory in clear text. The **secretEncryption** option is also used to encrypt pending updates while they are stored in the replication queue. Example: `replicaCredentials: secret`

Table 63. Replica entry schema definition (mandatory attributes) (continued)

Attribute	Description and example
cn	Forms the RDN of the LDAP distinguished name of the **replicaObject** entry. Example: `cn: myReplica`

In the examples in Table 63 on page 472, 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 root administrator DN and basic replication replica server DN and passwords" on page 151 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 might better describe the replica server, and it might listen on a different port than the default port of 389. Examples of adding a description and changing the port to 400 are shown in Table 64, which describes the optional attributes of **replicaObject**.

Table 64. Replica entry 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`
description	Provides an additional text field for extra information pertaining to the replica entry. Example: `description: Replica machine in the fourth floor lab`
seeAlso	Identifies another directory server entry that might 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`

Basic replication only supports simple authentication. SASL EXTERNAL, GSSAPI, DIGEST-MD5, and CRAM-MD5 bind mechanisms are not supported as valid basic replication bind mechanisms.

There are several additional attributes that affect error handling during basic replication. See "Basic replication error log" on page 483 for more information about error handling. These attributes are not in any object class, therefore, the **extensibleObject** object class must be included in the replica entry when adding these attributes to the entry. Table 65 describes these attributes.

Table 65. Additional optional replication attributes

Attribute	Description and example
ibm-slapdLog	Specifies the file name of the basic replication error log. This must be a UNIX System Services file (not a data set). The file name can be fully qualified or can be relative to the current working directory of the LDAP server. The current working directory is set when the LDAP server is started to the **HOME** environment variable if specified, or else to `/etc/ldap`. This format is not preferred. The value must be unique among all the replica entries in this LDAP server. If this attribute is not present in the replica entry or it has no value, error logging and setting aside does not occur. Example: `ibm-slapdLog: /home/replog/replica1.errlog`
ibm-slapdReplMaxErrors	Specifies the maximum number of basic replication errors that set aside in the basic replication error log before replication is allowed to stall. If this attribute is not present in the replica entry or if the value is 0, then no operations are set aside. In this case, errors are still logged and basic replication stalls when the first error occurs. This attribute is not used if a replication log file name has not been specified with the **ibm-slapdLog** attribute. Example: `ibm-slapdReplMaxErrors: 5`

Adding replica entries in TDBM or LDBM

In TDBM or LDBM, replica entries can be placed anywhere within the directory tree, although it is suggested that a replica entry be a leaf entry. Placing replica entries in the directory tree then requires that any parent entries of the replica entry be added to the directory before adding the replica entry. These entries must be added to both the replicating server and replica server before addition of the replica entry. This is needed on the replica server because these entries are being added at the replicating server without replication being active. If a replica entry is not placed as a leaf node in the directory tree, the only entries allowed below the replica entry are other replica entries. The LDAP server allows non-replica entries to be placed below replica entries, however, these entries are not replicated to the replica servers.

The replica entry defines a replica for the backend containing the entry. Any changes made to the directory tree managed by that backend is replicated to each replica defined for that backend. The replica entry does not define replicas for other backends in the LDAP server, therefore, if changes to all LDBM and TDBM

directory trees managed by the LDAP server are to be replicated, then each backend must contain the appropriate replica entries to define replication for that backend.

The following is an example of a replica entry definition using LDIF format.

```
dn: cn=myReplica,o=Your Company
objectclass: replicaObject
objectclass: extensibleObject
cn: myReplica
replicaHost: myMachine.ibm.com
replicaBindDn: cn=Master
replicaCredentials: secret
replicaPort: 400
replicaUseSSL: FALSE
description: Replica machine in the fourth floor lab
ibm-slapdLog: ro1.errlog
ibm-slapdReplMaxErrors: 5
```

Searching a replica entry

Most of the attributes in a replica entry are operational attributes. When searching a replica entry, the operational attributes are not included in the output unless they or the special + attributes are specified in the attributes to be returned. The following command searches for all replica entries in a suffix and returns the complete replica entries in LDIF format:

```
ldapsearch -h ldaphost -p ldapport -D binddn -w passwd -L -b "suffix"
 "objectclass=replicaObject" "*" replicaHost replicaBindDN replicaCredentials
 replicaPort replicaUpdateTimeInterval replicaUseSSL replicaBindMethod
```

Displaying basic replication status

The LDAP server **DISPLAY REPLICAS** operator modify command can be used to display information about the status of replication to each replica server. See “Displaying performance information and server settings” on page 184 for a description of the **DISPLAY REPLICAS** output.

Basic replication maintenance mode

Maintenance mode is the LDAP server setup mode for basic replication. This mode restricts access to the backends in an LDAP server to allow replica backends to be primed for basic replication. Access to the backends is as follows:

- read-only replica backend: The **masterServerDN** for the replica and an LDAP root administrator (for example, the **adminDN**) have unrestricted access.
- peer replica backend: The **peerServerDN** for the replica and an LDAP root administrator (for example, the **adminDN**) have unrestricted access.
- non-replica backends (including the schema entry): An LDAP root administrator (for example, the **adminDN**) have unrestricted access. The **masterServerDN** and **peerServerDN** have no access outside of the backends that specify them.

Other users can bind to the LDAP server, but cannot access any entries within the server.

ACL checking is performed during search operations from **masterServerDN** and **peerServerDN** but not during update and compare operations. Generally, ACLs do not apply when bound as an LDAP administrator, however, ACL filters may or may not reduce administrative rights. In addition, an LDAP root administrator has the capability in maintenance mode to modify attributes that are read-only and are

typically only set by the LDAP server, such as **ibm-entryuuid**. See Chapter 9, "Administrative group and roles," on page 159 for more information about administrative group and roles.

Note: The LDAP server schema entry is not part of any replica backend. When the LDAP server is not in maintenance mode, **masterServerDN** and **peerServerDN** can only update the LDAP server schema if the schema entry ACL permits them to. An LDAP root or schema administrator can always update the schema.

Pending replication entries are replicated to the other replica servers, but updates performed when in maintenance mode are not replicated.

Specify the **-m** option on the server startup command to start the LDAP server in maintenance mode.

The LDAP server **MAINTMODE** 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 dssrv,maintmode state
```

where *state* can be **on** to turn maintenance mode on or **off** to turn maintenance mode off (and turn on normal mode).

Replica server

Initialization, or population, of a replica directory requires several steps.

With basic replication, changes to the LDAP server schema entry on the replicating server are not replicated. A separate update of the LDAP server schema on the replica is required each time the schema is updated on the replicating server.

Replica servers must support the LDAP Version 3 protocol.

Populating a replica

1. Either start the replica and replicating servers in maintenance mode or use the LDAP server **MAINTMODE ON** operator modify command on each of these LDAP servers to put these servers into maintenance mode.
2. Unload the replicating server's directory contents if there are any entries. For TDBM or LDBM, use the **ds2ldif** utility (see "ds2ldif utility" on page 225).
3. Ensure that 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, the schema can be unloaded from the replicating server using **ds2ldif** and reloaded into the replica by an LDAP root or schema administrator with the **ldapmodify** utility.
4. Using a root administrator user, run the **ldapadd** utility to add a single replica entry into the backend directory on the replicating server to identify the new replica being populated.

   Note that in order to load the replica 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, go to step 8 on page 477.

6. Transport the LDIF file created in step 2 on page 476 to the replica server's location.
7. Load the LDIF file from 6 into the replica server. This can be done using an LDAP root administrator to run the **ldapadd** utility to load the LDIF file. Alternatively for TDBM, the replica server can be stopped and then the **ldif2ds** utility used to load the LDIF file.
8. Configure the replica (see next section).
9. Stop the replica server (if it is running) and then restart it in maintenance mode. If it contains a replica entry that defines this server as a replica of itself, use an LDAP root administrator to run the **ldapdelete** utility to remove that entry.
10. Use the LDAP server **MAINTMODE OFF** command on the replica server and the replicating server to change these servers to normal mode.

Configuring the replica

The key to a successful replica configuration rests in ensuring that the values in the replica entry 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 LDAP server 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

The following table identifies the relationship between the attributes in the replica entry 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 **replicaHost** must specify either the corresponding host name or an IP address of one of the addresses.

Attribute in replica entry on replicating server	**Corresponding replica server configuration option or command line parameter**
replicaHost	The host name 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

Attribute in replica entry on replicating server	Corresponding replica server configuration option or command line parameter

Note:

1. 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.
2. The value of the **listen** configuration option or **–l** command line parameter is an LDAP URL. For additional information about the **listen** option, see Chapter 8, "Customizing the LDAP server configuration," on page 83.
3. It is preferred that the **masterServerDN** or **peerServerDN** be a DN that is dedicated specifically to replication. It should not be used for any other operations.
4. The **masterServer**, **masterServerDN**, **masterServerPW**, **peerServerDN**, and **peerServerPW** options must follow the **database** option for that backend in the LDAP server configuration file.
5. Usage of the **masterServerPW** or **peerServerPW** configuration option is strongly discouraged in production environments. See "Establishing the root administrator DN and basic replication replica server DN and passwords" on page 151 for alternatives.
6. The **replicaCredentials** attribute is encrypted if the **secretEncryption** configuration option is specified. This improves directory security because the bind password is no longer stored in the directory in clear text. The **secretEncryption** configuration option is also used to encrypt pending updates while they are stored in the replication queue.

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 replica server returns a referral containing the value in the **masterServer** option in the backend section of the LDAP server configuration file on the replica. The client then redirects the request to the master server. After the master server makes the update, it propagates the change to the read-only replica server, binding as the **replicaBindDn** value in the replica entry corresponding to that replica server (the **replicaBindDn** value must match the **masterServerDN** value in the replica server configuration file).

See "SSL/TLS and basic replication" on page 482 for information about securing a directory.

Changing a read-only replica to a master

When using read-only basic replication, you might want 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 you want 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. If the read-only replica is out of sync with the master server, use the procedure that is described in "Recovering from basic replication out-of-sync conditions" on page 485.
2. Use the LDAP server **MAINTMODE ON** operator modify command on the master server and on the replica server to put them into maintenance mode.

3. Using a root administrator DN, unload all the replica entries (entries that describe replica servers) from the master server. Use a search command, like the one shown in "Searching a replica entry" on page 475, to create LDIF output containing the replica entries for each suffix in the backend. In the LDIF output, remove the replica entry for the read-only replica that is going to become the master. If the master is going to become a read-only replica, add a replica entry for the master in LDIF format to the output.
4. Using a root administrator DN, run **ldapdelete** to remove the replica entries from the master.
5. Using a root administrator DN, run **ldapadd** to add the unloaded replica entries to the replica server.
6. Stop the master and replica server.
7. Remove the **masterServer, masterServerDN**, and **masterServerPW** options from the LDAP server configuration file on the replica.
8. If the original master is being eliminated, the database on the master is no longer needed.
   - For TDBM, drop the TDBM DB2 database. The SPUFI script that is used to create the DB2 database also contains the commands to drop the database.
   - For LDBM, remove all the files in the LDBM database directory. See the description of the **databaseDirectory** option in Chapter 8, "Customizing the LDAP server configuration," on page 83 for more information about the location of these files.
9. If the original master is going to become a replica, add the **masterServer, masterServerDN**, and **masterServerPW** options to the LDAP server configuration file on the original master. The **masterServer** value must point to the new master. See "Establishing the root administrator DN and basic replication replica server DN and passwords" on page 151 for more information about alternatives to specifying the **masterServerPW** option.
10. Start the new master server and new replica server (if the original master became a replica server).

Basic peer to peer replication

z/OS LDAP peer replication server provides failover support. With this support, if an 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: Basic peer to peer replication uses the same replica entry attribute values as shown in "Replica server" on page 476. The instructions in "Adding replica entries in TDBM or LDBM" on page 474 also apply to peer replicas.

A basic 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

The **peerServerDN** and **peerServerPW** options in the backend section of the LDAP server configuration file are used to configure a basic peer to peer replication environment. See Chapter 8, "Customizing the LDAP server configuration," on page 83 for more information.

Note: Usage of the **peerServerPW** configuration option is strongly discouraged in production environments. See "Establishing the root administrator DN and basic replication replica server DN and passwords" on page 151 for alternatives.

Basic replication conflict resolution

Minimal conflict resolution is done in a basic 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, basic replication can stall on server A. Ensure that your peer servers are not receiving conflicting operations. To avoid basic replication stalling, set up a replication error log to set aside replication errors. See "Basic replication error log" on page 483 for more information.

When a conflict occurs, a notification is sent to the console and server log.

Adding a peer replica to an existing server

For failover support, it might be necessary for you to add a peer replica for a backend 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 for a backend 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 backend section of the LDAP server configuration file.
2. Stop the existing servers. For each existing server that is to become a peer server, update its configuration file to include the **peerServerDN** and **peerServerPW** configuration options. Restart the existing read/write servers in maintenance mode. See "Establishing the root administrator DN and basic replication replica server DN and passwords" on page 151 for alternatives to specifying the password in the configuration file.
3. Prime the new peer replica with all the data from an existing server. You can accomplish this by dumping the existing server's directory (for TDBM or LDBM, use **ds2ldif**) and adding the data to the new peer replica (for TDBM or LDBM use **ldapadd** or, for TDBM, use **ldif2ds**). See "Populating a replica" on page 476 for more information.
4. Add a replica entry to the existing servers to point to the new peer replica.
5. Add a replica entry 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 entries defined to it, those replica entries might have been copied to the new peer replica in step 3 above. Ensure that this server does not contain a replica entry 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 might be necessary for you to upgrade a read-only replica for a backend 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 for a backend to a peer replica:

1. Stop both the master server and the read-only replica.
2. Remove the **masterServer, masterServerDN**, and **masterServerPW** options from the backend section of the LDAP server configuration file of the read-only replica.
3. Add a **peerServerDN** and **peerServerPW** option to the backend section of each server's configuration file. The two servers are now peer servers. See "Establishing the root administrator DN and basic replication replica server DN and passwords" on page 151 for alternatives to specifying the password in the configuration file
4. Start both servers in maintenance mode.
5. In this backend, on the read-only replica being upgraded:
   - Add a replica entry for each replica that this backend on the master server points to (except the entry 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 entry to point to the master.
6. On the master, ensure that the credentials are valid in the replica entry 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 for the backend.

Downgrading a peer server to read-only replica

It might be necessary for you to downgrade a backend from 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 backend section of the LDAP server configuration file.
3. Add **masterServer, masterServerDN**, and **masterServerPW** options to the backend section of the peer replica configuration file. If there are more than one peers, add a **masterServer** option for each one. See "Establishing the root administrator DN and basic replication replica server DN and passwords" on page 151 for alternatives to specifying the password in the configuration file.
4. Ensure that the credentials are valid in the replica entry for the newly downgraded peer server on all the replicating servers.
5. Start the server.

The peer server is now a read-only replica for the backend.

SSL/TLS and basic 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 like a typical SSL/TLS server. It needs its own public-private key pair and certificate, and the LDAP server configuration file needs the standard SSL options (**listen, sslKeyRingFile**, and **sslKeyRingFilePW**). See "Setting up for SSL/TLS" on page 68 for more information.

Replicating server with SSL/TLS enablement

The replicating server acts as 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 SSL Programming*) or the RACDCERT command (see *z/OS Security Server RACF Command Language Reference*), this time as if you were the client. The key database file, RACF keyring, or PKCS #11 token must contain the replicating server's key pair and certificate. Receive the replica's self-signed certificate and mark it as trusted.
2. In the LDAP server configuration file on the replicating server:
   - Set **sslKeyRingFile** to the replica key database file, RACF keyring, or PKCS #11 token created above.
   - If a replica key database file is used, set **sslKeyRingFilePW** to the password for the key database file, or set **sslKeyRingPWStashFile** to the file name where the password is stashed.
3. Ensure any environment variables that control SSL/TLS settings are properly defined in the LDAP server environment variable file. The environment variables for enabling TLS protocol levels are shared with the server definitions. For example, GSK_PROTOCOL_TLSV1_2=ON enables this protocol level for both inbound client connections to the replicating server and for outbound connections from the replicating server to the replica. However, since the replicating server acts as an SSL/TLS client to the replica, the environment variable usage for controlling cipher suites is as described for the client API. The SSL cipher format you should use on the outbound connections to the replicas is controlled by the LDAP_SSL_CIPHER_FORMAT environment variable and then either the GSK_V3_CIPHER_SPECS or GSK_V3_CIPHER_SPECS_EXPANDED environment variable, depending on which format is chosen. The SSL cipher suites on the inbound client connections are controlled by the configured setting of sslCipherSpecs and can potentially share the setting that is specified on GSK_V3_CIPHER_SPECS_EXPANDED. Where settings are shared for both inbound client connections and outbound connections to replicas, the cipher list must include the necessary cipher suites for both sets of connections.
4. In the replica entry for this replica:
   - Set the **replicaPort** attribute to the replica's secure port number.
   - Set the **replicaUseSSL** attribute to **TRUE**.

See "Setting up for SSL/TLS" on page 68 for more information.

Because the replicating server acts as an SSL/TLS client to the replica server, the replicating server binds with the replica server. The bind method that is used is **simple** bind. The SASL external bind method is not supported for basic replication.

Basic replication error log

A replication error log holds information about each error that occurs during basic replication. To avoid stalling basic replication, the failed replication operation is taken off the replication queue so that replication can continue with the next operation. Depending on the error, the LDIF of the failed operation is set aside (added) to the error log.

There is one error log for each replica of a backend. The file name of the error log for a replica is specified by the **ibm-slapdLog** attribute in the replica entry for that replica within the backend. The file name must be unique across the LDAP server. If the attribute does not exist in the replica entry or the attribute has no value, no errors are logged or replication operations set aside during this backend's replication to that replica. In this case, basic replication to that replica stalls every time a failure occurs. The **ibm-slapdReplMaxErrors** attribute in the replica entry is set to control how many failed replication operations can be set aside each time the LDAP server is started before basic replication stalls for that replica.

In a sysplex environment, consider having the basic replication error log file shared by the LDAP servers which are sharing the replicating backend. Only the LDAP server that is the owner of the backend in the sysplex performs replication and writes to the replication error log if an error occurs. However, if that LDAP server stops, another LDAP server in the sysplex becomes owner and handles basic replication. If the error log file is shared, then the new owner can add error information to the same error log rather than beginning a new error log. This centralizes error information and eliminates the need to look for error logs on each server that might have become the backend owner.

The replication error log is used to correct basic replication in two ways:

- Use the error information to determine why replication failed.
- Use the **ldapmodify** utility to run the error log on the replica server, after resolving the basic replication problems. This performs the modifications that were set aside in the error log, therefore, bringing the backend in the replica to the same level as in the replicating server. You must bind as either the **masterserverDN** or **peerserverDN**, depending on the type of replica.

The following is an example of an error log entry:

```
#(051219 13:25:23.146587): modify operation failed for cn=IBMUSER01, o=ibm,c=us to
  dceimgtd.pdl.pok.ibm.com:390, rc=32
# R004071 Entry cn=IBMUSER01, o=ibm,c=us  does not exist
  (tdbm_process_request:933)
# setting change aside.

dn: cn=IBMUSER01, o=ibm,c=us
changetype: modify
modify: uid
uid: IBMUSER1
-
modify: userpassword
userpassword:: Ym9i
```

The basic replication error log consists of three messages, each using one or more lines:

1. Message one indicates when the error occurred, the entry, and replica server.
2. Message two is the error message returned by the replica server.

3. Message three indicates what is being done. If the operation is set aside, this message is followed by the LDIF of the operation.

All non-LDIF information is prefixed with the comment character # so that the error log can be run through **ldapmodify** to synchronize the two servers.

Following is an example in which a replication error condition is logged but no set-aside of the modification is needed:

```
#(051219 14:13:12.366227): delete operation failed for cn=IBMUSER01,
o=ibm,c=us to dceimgtd.pdl.pok.ibm.com:390, rc=32
# R004071 Entry cn=IBMUSER01, o=ibm,c=us does not exist
 (tdbm_process_request:933)
# Entry is already deleted, ignoring request.
```

There is no LDIF. Notice the third message indicates that request is being ignored.

Troubleshooting basic replication

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 a replica entry for the replica server exists in the backend to be replicated in the replicating server, and was specified correctly to match with the replica server. If **cn=localhost** is used as the suffix for all replica entries for a backend, 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 backend section of the LDAP server configuration file and the filter is **objectClass=replicaObject**. If more than one suffix is configured for TDBM or LDBM, the search must be repeated using each suffix in the search base.

  See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information about **ldapsearch**.
- Verify that the **replicaHost** value in the replica entry for that replica specifies the machine on which the replica is running.
- Check that the values listed in the replica entry 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 entry 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 LDAP server configuration file in the replica server points to the replicating server.
- If the replica entry **replicaUseSSL** attribute is set to **TRUE**, verify the **replicaPort** attribute is set to the SSL port configured on the replica server. Verify the **sslKeyRingFile**, and **sslKeyRingFilePW** or **sslKeyRingPWStashFile** values in the LDAP server configuration file on the replica server and on the replicating server are correct.
- When adding many 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 twenty times the size of the largest LDIF file that is to be added to the replicating server.

The **ibm-slapdLog** and **ibm-slapdReplMaxErrors** attributes in a replica entry can be used to configure a replication error log for this replica. If basic replication fails, the error log holds all errors that occurred during replication and the LDIF for the set aside replication operations.

Recovering from basic replication 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 might be necessary to reload the replica.

The following procedure should be followed to reload a replica:

1. Use the LDAP server **MAINTMODE ON** operator modify command on the replicating sever and on each of the replica servers to put them into maintenance mode.
2. Using an root administrator DN, unload all the replica entries (entries that describe replica servers) from the master server. Use a search command like the one shown in "Searching a replica entry" on page 475 to create LDIF output containing the replica entries for each suffix in the backend.
3. Using an root administrator DN, run **ldapdelete** to remove the replica entries from the master. This resets the replication information in the replicating server.
4. For TDBM, run the following SPUFI on the replicating server to be sure that the server successfully completed the removal of the data in the DIR_REPLICA, DIR_REPENTRY and DIR_LONGREPENTRY tables:

   ```
   select count(*) from dbuserid.dir_replica
   ```

   where you substitute your database owner for *dbuserid*. The record count returned should be zero.
5. Stop all the replica servers.
6. Clear out the directory on each replica server.
   - For TDBM, drop and re-create the TDBM DB2 database. See "Creating the DB2 database and table spaces for TDBM or GDBM" on page 55 for an example of the SPUFI commands needed to do this.
   - For LDBM, remove all the files in the LDBM database directory. See the description of the **databaseDirectory** option in Chapter 8, "Customizing the LDAP server configuration," on page 83 for more information about the location of these files.
7. Run an unload utility on the replicating server. Use **ds2ldif** twice, once to unload the schema entry and a second time to unload the TDBM or LDBM directory entries.
8. Start the replica servers in maintenance mode.
9. Using an administrator DN, run **ldapmodify** to load the schema unloaded from the replicating server onto each replica.
10. On each replica, load the directory data retrieved above from the replicating server. For LDBM, you must use **ldapadd**. For TDBM, you can use **ldapadd** or use the **ldif2ds** load utility. The **ldapadd** utility must be run using an root

administrator. If you use **ldif2ds**, you must stop the replica server before loading entries. In this case, restart the replica in maintenance mode after loading it.

11. Using an administrator DN, run **ldapadd** to add the replica entries unloaded in step 4 on page 476 back into the replicating server.
12. Use the LDAP server **MAINTMODE OFF** operator modify command to take the replicating server and each replica out of maintenance mode.

Chapter 26. Advanced replication

Replication keeps data in multiple directory servers synchronized. Advanced replication includes the following function:

- Allows specific subtrees within the Directory Information Tree (DIT) to be chosen for participation in advanced replication topologies (in contrast to requiring the entire backend to participate or not participate)
- Allows the subtrees participating in an advanced replication environment to have different roles (for example, supplier or consumer)
- Additional replication topology choices can be combined to serve many different directory information architectures and data redundancy requirements
- External error log management using extended operations
- Operational attributes to determine the current state of the advanced replication environment
- External queue management using extended operations
- Password policy updates
- Schema replication

Advanced replication terminology

Cascading replication
A replication topology with multiple tiers of servers. A peer-master server replicates to a small set of read-only servers that replicate to other servers. Such a topology offloads replication work from the master servers.

Consumer server
A server that receives changes from replication from another (supplier) server.

Credentials entry
An entry that identifies the method and required information that the supplier uses in binding to the consumer. For simple binds, it is the distinguished name (DN) and password. This entry is specified in the replication agreement.

Forwarding server
A read-only server that replicates all changes sent to it. This contrasts to a peer-master server in that a peer-master server does not replicate changes sent to it from another peer-master server; it only replicates changes that are originally made on the peer-master server.

Gateway server
A server that forwards all replication traffic from the local replication site where it exists to other gateway servers in the replicating network. This server also receives replication traffic from other gateway servers within the replication network, that it forwards to all servers on its local replication site. Gateway servers must be masters (writable).

Master server
A server that is writable (can be updated) for a given subtree.

Nested subtree
A subtree within another subtree of the directory.

Peer server
The term used for a master server when there are multiple masters for a given subtree. A peer server does not replicate changes sent to it from another peer server; it only replicates changes that are originally made on it.

Replica group
The first entry created under a replication context has objectclass **ibm-replicaGroup** and represents a collection of servers participating in replication. It provides a convenient location to set ACLs to protect the replication topology information.

Replica subentry
Below a replica group entry, one or more entries with objectclass **ibm-replicaSubentry** can be created; one for each server participating in replication as a supplier. The replica subentry identifies the role the server plays in replication: master or read-only. A read-only server might, in turn, have replication agreements to support cascading replication.

Replicated subtree
A portion of the Directory Information Tree (DIT) that is replicated from one server to another. Under this design, a given subtree can be replicated to some servers and not to others. A subtree can be writable on a given server, while other subtrees might be read-only.

Replicating network
A network that contains connected replication sites.

Replication agreement
Information contained in the directory that defines the "connection" or "replication path" between two servers. One server is called the supplier (the one that sends the changes) and the other is the consumer (the one that receives the changes). The agreement contains all the information needed for making a connection from the supplier to the consumer and scheduling replication.

Replication context
Identifies a portion of the Directory Information Tree (DIT) that is allowed to be replicated from one server to another. The **ibm-replicationContext** auxiliary object class might be added to an entry to mark it as the root of a replicated area. The configuration information related to replication is maintained in a set of entries created below the base of a replication context.

Replication filter
An entry containing the list of attributes that must be replicated or excluded from replication corresponding to a particular type of entry. It can exist anywhere in the Directory Information Tree (DIT) but is always associated with an agreement.

Replication site
A gateway server and any master, peer, or replica servers configured to replicate together.

Replication topology
The set of entries in a directory that control the type of information that is replicated between LDAP servers and how it is replicated. These objects include:

- Replica groups
- Replica subentries

- Replication agreements
- Replication contexts
- Replication credentials entries
- Replication schedule entries

All LDAP servers in the replicating network must have the same replication topology.

Replication schedule
Replication can be scheduled to occur at particular times, with changes on the supplier accumulated and sent in a batch. The replication agreement contains the distinguished name (DN) for the entry that supplies the schedule.

Supplier server
A server that sends changes to another (consumer) server.

Replication topology

When advanced replication is configured, specific entries in the directory are identified as the roots of replication subtrees or replication contexts by adding the auxiliary objectclass **ibm-replicationContext** to them. Each of these replication contexts are replicated independently. The subtree continues down through the Directory Information Tree (DIT) until reaching the leaf entries or other replicated subtrees or contexts. Entries are added below the root of the replicated subtree to contain the replication configuration information. There are one or more replica group entries created directly under each replication context. For each replica group entry, there is a corresponding replica subentry that identifies the role the server plays in the replication environment. Associated with each replica subentry are replication agreements that identify the servers that are supplied (replicated to) by each server and defining the credentials and schedule information.

By using advanced replication, a change made to one server is propagated to one or more additional servers. In effect, a change to one server can show up on multiple different LDAP servers. z/OS IBM Tivoli Directory Server supports either basic or advanced replication but not both at the same time. Advanced replication includes:

- replication of subtrees of the Directory Information Tree to specific servers
- a multitier topology, referred to as cascading replication
- assignment of server role (supplier or consumer) by subtree
- multiple master servers, referred to as peer to peer replication
- gateway servers that replicate across networks

The advantage of replicating by subtrees is that a replica does not need to replicate the entire directory. It can be a replica of a part, or subtree, of the directory.

The advanced replication model changes the concept of master and replica servers. These terms no longer apply to servers, but rather to the roles that a server has regarding a particular replicated subtree. A server can act as a master for some subtrees and as a replica for others. The term, master, is used for a server that accepts client updates for a replicated subtree. The term, replica, is used for a server that only accepts updates from other servers designated as a supplier for the replicated subtree.

The types of directory roles as defined by function are: master-replica, peer-peer, forwarding (cascading), gateway

Table 66. Server roles

Option	Description
Master-replica	A replica is an additional server that contains a copy of the directory information that is replicated from the master server. The replicated data can be the entire DIT or just a portion of the DIT that is replicated to the replica. The replica server provides a read-only backup of the replicated subtree.
Master-peer	The master-peer server contains the master directory information from where updates are propagated to the replicas. All changes are made and occur on the master server, and the master is responsible for propagating these changes to the replicas. There can be several servers acting as masters for directory information, with each master responsible for updating other master servers and replica servers. This is referred to as peer replication. Peer replication can improve performance and reliability. Performance is improved by providing a local server to handle updates in a widely distributed network. Reliability is improved by providing a backup master server ready to take over immediately if the primary master fails. **Note:** 1. Master servers replicate all client updates, but do not replicate updates received from other masters. 2. Updates among peer servers can be immediate or scheduled.
Forwarding (Cascading)	A forwarding or cascading server is a replica server that replicates all changes sent to it. This contrasts to a master-peer server in that a master-peer server only replicates changes that are made by clients connected to that server. A cascading server can relieve the replication workload from the master servers in a network that contains many widely dispersed replicas.
Gateway	Gateway replication uses gateway servers to collect and distribute replication information effectively across a replicating network. The primary benefit of gateway replication is the reduction of network traffic.

You can request updates on a replica server, but, the update is forwarded to the master server by returning a referral to the client. If the update is successful, the master server then sends the update to the replicas. Until the master has completed replication of the update, the change is not reflected on the replica server where it was originally requested. If replication fails, it is repeated even if the master is restarted. Changes are replicated in the order that they are made on the master. See “Recovering from advanced replication errors” on page 547 for more information.

If you are no longer using a replica, you must remove the replication agreement entry from the supplier. Leaving the entry causes the server to queue up all updates and uses unnecessary directory space. Also, the supplier continues trying to contact the missing consumer to try sending the data again. When a replication agreement is deleted, replication is halted immediately. That is, any updates in the replication queue are lost.

Advanced replication overview

This section presents a high-level description of the various advanced replication topologies.

Master-replica replication

The basic relationship in advanced replication is that of a master server and its replica server. The master server can contain a directory or a subtree of a directory. The master is writable, and means it can receive updates from clients for a given subtree. The replica server contains a copy of the directory or a copy of part of the directory of the master server. The replica is read only; it cannot be directly updated by clients. Instead it refers client requests to the master server, that performs the updates and then replicates them to the replica server.

A master server can have several replicas. Each replica can contain a copy of the masters entire directory, or a subtree of the directory. In the following example, `Replica 2` contains a copy of the complete directory of the `Master Server`, `Replica 1`, and `Replica 3` each contain a copy of a subtree of the `Master Server's` directory.

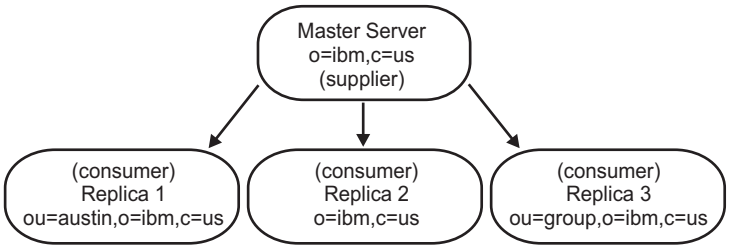


Figure 45. Master-replica replication

The relationship between two servers can also be described in terms of roles, either supplier or consumer. In the previous example, the `Master Server` is a supplier to each of the replicas. Each replica in turn is a consumer of the `Master Server`.

Forwarding (cascading) replication

Forwarding (cascading) replication is a topology that has multiple tiers of servers. A master server replicates to a set of read-only (forwarding) servers that in turn replicate to other servers. Such a topology offloads replication work from the master server. In the example of this type of topology, the master server is a supplier to the two forwarding servers. The forwarding servers serve two roles. They are consumers of the master server and suppliers to the replica servers associated with them. The replica servers are consumers of their respective forwarding servers. For example:

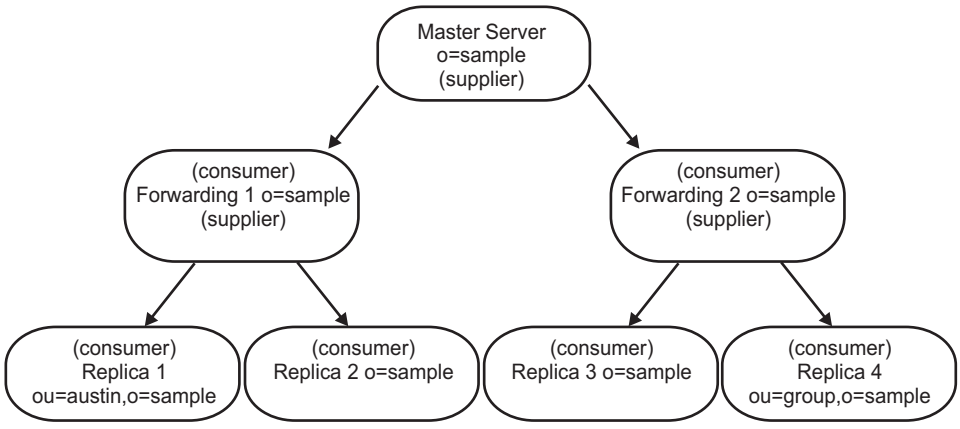


Figure 46. Cascading replication

Peer-to-peer replication

There can be several servers acting as masters for directory information, with each master responsible for updating other master servers and replica servers. This is referred to as peer replication. Peer replication can improve performance, availability, and reliability. Performance is improved by providing a local server to handle updates in a widely distributed network. Availability and reliability are improved by providing a backup master server ready to take over immediately if the primary master fails. Peer master servers replicate all client updates to the replicas and to the other peer masters, but do not replicate updates received from other master servers.

Note: Conflict resolution for add and modify operations in peer-to-peer replication is based on timestamps of entries. See "Replication conflict resolution" on page 493 for more information.

Figure 47 is an example of peer-to-peer replication:

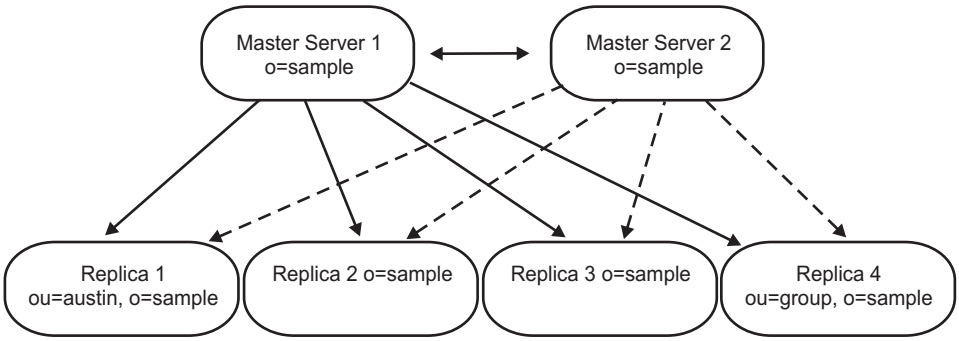


Figure 47. Peer-to-peer replication

Gateway replication

Gateway replication is a more complex adaptation of peer-to-peer replication that extends replication capabilities across networks. The most notable difference is that a gateway server does replicate changes received from other peer servers through the gateway.

A gateway server must be a master server, that is, writable. It acts as a peer server within its own replication site. That is, it can receive and replicate client updates and receive updates from the other peer-master servers within the replication site. It does not replicate the updates received from the other peer-masters to any servers within its own site.

Within the gateway network, the gateway server acts as a two-way forwarding server. In one instance, the peers in its replication site act as the suppliers to the gateway server and the other gateway servers are its consumers. In the other instance, the situation is reversed. The other gateway servers act as suppliers to the gateway server and the other servers within its own replication site are the consumers.

Gateway replication uses gateway servers to collect and distribute replication information effectively across a replicating network. The primary benefit of gateway replication is the reduction of network traffic. For example:

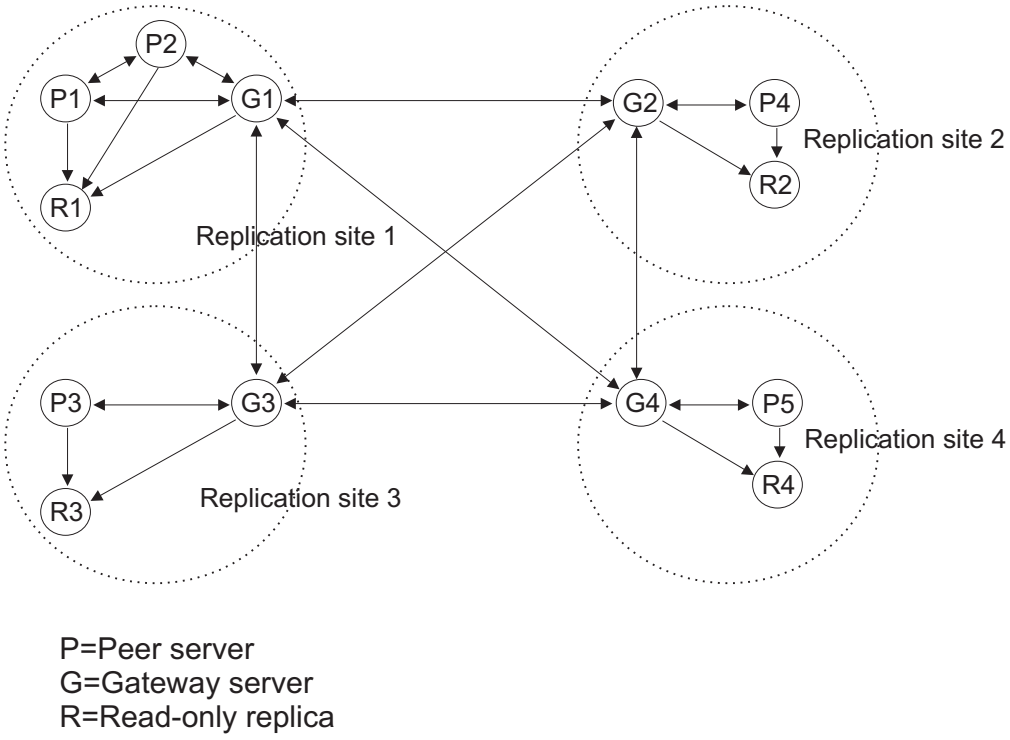


Figure 48. Gateway replication

Advanced replication features

This topic presents a high-level overview of advanced replication features.

Partial replication

Partial replication is an advanced replication feature that replicates only the specified entries and a subset of attributes for the specified entries within a subtree. Using partial replication, an LDAP administrator can enhance the replication bandwidth depending on the deployment requirements. The attributes that are to be replicated are specified using a replication filter. For more information about partial replication, see "Partial replication" on page 536.

Replication scheduling

Replication scheduling is an advanced replication feature that allows updates to be queued and then replicated at a certain time each day or during certain days of the week. Using replication scheduling, an LDAP administrator can schedule advanced replication to occur at optimal times when network traffic is minimal. For more information about replication schedule entries, see "Schedule entries" on page 503.

Replication conflict resolution

If there are replication conflicts involving delete or modifyDN operations, LDAP administrator intervention might be needed to correct the problems. For example, if an entry is renamed on one server while it is being modified on a second server, the modifyDN might arrive at a replica before the modify. Then when the modify arrives, it fails. In this case, an administrator needs to respond to the error by applying the modify to the entry with the new distinguished name (DN). All

information necessary to redo the modify with the correct name is preserved in the replication and error logs. Replication errors are rare occurrences in a correctly configured replication topology, but it is not safe to assume that they never occur.

Conflict resolution for add and modify operations in peer-to-peer replication is based on the **modifyTimeStamp** attribute value. The entry update with the most recent **modifyTimestamp** on any server in a multi-master replication environment is the one that takes precedence. Replicated delete and rename (modify DN) requests are accepted in the order received without conflict resolution. When a replication conflict is detected, the replaced entry is archived for recovery purposes in the lost and found log that is specified in the **ibm-slapdLog** attribute of the **cn=Replication,cn=Log Management,cn=configuration** entry.

Updates to the same entry made by multiple servers might cause inconsistencies in directory data because conflict resolution is based on the **modifyTimestamp** value of the entries. The most recent **modifyTimestamp** value takes precedence. If the data on your servers becomes inconsistent, use the synchronization procedure in "Recovering from advanced replication errors" on page 547 to resynchronize the servers.

For advanced replication conflict resolution to work correctly, the supplier server must provide the modified entry's **modifyTimestamp** value before the entry was updated on the supplier. The consumer server uses the **modifyTimestamp** attribute value to determine what to do with a modified entry. If the consumer server receives a **modifyTimestamp** value on an entry that is earlier than the same entry's **modifyTimestamp** in its own server, then the modify request from the supplier server is ignored. However, this same replication conflict resolution does not occur for the schema entry, **cn=schema**. The replicated **cn=schema** entry is always replaced on the consumer server even if the consumer server has a later **modifyTimestamp** value.

Enabling advanced replication

Before advanced replication entries are allowed to be added to the TDBM or LDBM backends, the CDBM backend must be configured in the LDAP server configuration file and the **useAdvancedReplication** configuration option set to **on** in the CDBM backend. For example:

```
database CDBM GLDBCD31/GLDBCD64
databaseDirectory /var/ldap/cdbm
useAdvancedReplication on
```

Note:

1. The CDBM backend is only allowed to be configured when the server compatibility is 5 or greater. See "serverCompatLevel {3 | 4 | 5 | 6 | 7| 8}" on page 125 for more information about the **serverCompatLevel** configuration option.
2. If **useAdvancedReplication on** is specified in the CDBM backend and basic replication entries with an objectclass of **replicaObject** exist in any configured TDBM or LDBM backends, the server does not start. Entries with an objectclass of **replicaObject** are not allowed to be added when advanced replication is allowed. Basic and advanced replication environments are not supported at the same time in the z/OS LDAP server. If planning to use an advanced replication environment, all basic replication **replicaObject** entries must be removed from the TDBM or LDBM backends.

3. If there are advanced replication entries in the LDBM and TDBM backends and **useAdvancedReplication off** is specified in the CDBM backend, the server does not start because basic replication is intended to be used. Replication contexts, replica groups, replica subentries, and replication agreement entries are not allowed to be added when basic replication is allowed.
4. The **masterServer, masterServerDN, masterServerPW, peerServerDN**, and **peerServerPW** configuration options are not allowed to be specified in any LDBM or TDBM backends when the CDBM backend is configured and the **useAdvancedReplication** option is set to **on**. The **masterServer, masterServerDN, masterServerPW, peerServerDN**, and **peerServerPW** options are only valid when the server is configured to run in a basic replication environment.

The **cn=configuration** suffix contains entries that are used to configure advanced replication support. When the server is first started, the following advanced replication configuration entries under the **cn=configuration** suffix are automatically created:

- **cn=configuration**
- **cn=Replication,cn=configuration**
- **cn=Log Management,cn=Configuration**
- **cn=Replication,cn=Log Management,cn=Configuration**

The **cn=localhost** suffix is a special suffix that is exempt from replication. It is not required, but allowed in any TDBM or LDBM backend. It is an appropriate location for adding supplier entries that do not need to be replicated, such as supplier server credential entries. To create entries under the **cn=localhost** suffix, you must define the suffix in the appropriate backend section of the LDAP server configuration file and also populate the suffix entry. The simplest entry to create is a container object, using the following LDIF:

```
dn: cn=localhost
objectclass: container
```

If advanced replication is being enabled for the first time for a TDBM backend, which was created before z/OS V1R11, the DB2 database must be updated to enable TDBM to be configured in an advanced replication environment. The changes are explained in section 3 of the TDBMMGRT member of the *GLDHLQ*.SGLDSAMP data set.

Follow these steps:

1. Copy the TDBMMGRT member to your own SPUFI input data set. Edit your version of TDBMMGRT. Read the commentary in section 3 to understand what this SPUFI script is going to do. You must replace -DB2_NAME-, -DB2_USERID-, -MISC_TABLESPACE-, -REPLICA_TABLESPACE-, and -STORAGEGROUP- with the appropriate values for the TDBM database you are migrating.
2. Stop the LDAP server.
3. Use the DB2 SPUFI facility to run your version of TDBMMGRT. The script must be run under a user ID with DB2 SYSADM authority. When the script completes running, scan the output to ensure that it ran successfully.
4. Start the LDAP server.

See "CDBM backend configuration and policy entries" on page 139 for more information about these entries and attribute values that affect advanced replication configuration.

Supplier server entries

The following sections indicate the entries that must be added or modified in the supplier server to successfully configure advanced replication.

Replication contexts

A replication context is an entry in the directory with the auxiliary objectclass **ibm-replicationContext** that identifies the root of a replicated subtree. The auxiliary objectclass **ibm-replicationContext** is allowed to be added to any entry in the directory. See Table 67 for the optional attribute value for the **ibm-replicationContext** objectclass. The replication configuration information is maintained in a set of entries created below the base of a replication context. If there is more than one replication context present in the same subtree, the replication configuration information under the child replication context entry is used while the replication configuration under the parent entry is ignored.

If the **ibm-replicationContext** auxiliary objectclass is added to a non-suffix level entry in the directory, explicit **aclEntry** and **entryOwner** attribute values are required. When an **entryOwner** attribute value is required, it must start with an *access-id:*. See "Protecting replication topology entries" on page 534 for more information about protecting the replication topology.

Table 67. **ibm-replicationContext** *objectclass schema definition (optional attribute)*

Attribute description and example
ibm-replicaReferralURL A single valued attribute that contains an ordered list of LDAP URLs with server name and optional port numbers that are separated by spaces. This list contains a list of servers that have update access to this replication context. Example: `ibm-replicaReferralURL: ldap://master1.ibm.com:500 ldaps://master2.ibm.com:636`

For the example in Table 67, assume that a replication context of `o=ibm` is used. When a client attempts an update operation on the consumer server under the `o=ibm` replication context, the referral list in the **ibm-replicaReferralURL** attribute value is sent back to the client indicating the supplier servers for the replication context.

Replica groups

A replica group entry is created directly under a replication context entry with the structural objectclass **ibm-replicaGroup**. See Table 68 on page 497 for the optional and required attributes for the **ibm-replicaGroup** objectclass. A replica group entry represents a collection of servers participating in replication for the context. Multiple replica group entries are allowed to be created under a replication context. A replica group entry provides a convenient location to set ACLs to protect the replication topology information, however, these entries do not do anything as far as how the replication topology is configured. See "Protecting replication topology entries" on page 534 for more information about protecting the replication topology.

Table 68. **ibm-replicaGroup** *objectclass schema definition (optional and required attributes)*

Attribute description and example
description An optional attribute that provides a text field for extra information pertaining to the replica group entry. This attribute does not affect advanced replication configuration. Example: `description: Replica group 1`
ibm-replicaGroup A required attribute value that specifies the name of a replica group. Example: `ibm-replicaGroup: Group1`

Replica subentries

A replica subentry is created directly under a replica group with the structural objectclass **ibm-replicaSubentry**. See Table 69 for the optional and required attributes for the **ibm-replicaSubentry** objectclass. A replica subentry identifies the role that the server plays in advanced replication (for example master, peer, forwarding, or gateway server). Do not create the replica subentry if the server's role is a read-only replica server. If the auxiliary objectclass **ibm-replicaGateway** is added to a replica subentry, the server's role is a gateway server. See "Gateway replication" on page 492 for more information. There should only be one replica subentry created for a single server under a given replication context.

Table 69. **ibm-replicaSubentry** *objectclass schema definition (optional and required attributes)*

Attribute description and example
cn A required attribute that specifies the common name of the replica subentry. This attribute does not affect advanced replication configuration. Example: `cn: Subentry 1`
description An optional attribute that provides an additional text field for extra information pertaining to the replica subentry. This attribute does not affect advanced replication configuration. Example: `description: Represents the LDAP server (master1) under this replication context`
ibm-replicaServerID A required attribute that specifies the server ID of the server that this entry represents. To determine the ID of a server, search the root DSE entry for the **ibm-serverID** attribute. This value must exactly match the **ibm-serverID** attribute value because it is case-sensitive. This attribute cannot be modified after this entry is created. If this attribute value must be changed, all entries under the replica subentry must be deleted and then added again. A server does not interrogate the replica subentry or any replication agreements beneath it when the **ibm-replicaServerID** attribute value does not match its own ID. See "Replication agreements" on page 498 for more information about replication agreement entries. Example: `ibm-replicaServerID: supplier1`

Table 69. **ibm-replicaSubentry** *objectclass schema definition (optional and required attributes) (continued)*

Attribute description and example
ibm-replicationServerIsMaster A required boolean (true or false) attribute that indicates whether the server represented by this replica subentry, as determined by the **ibm-replicaServerID** attribute value, is a master server for the replication context. If set to true, the server that is represented by this replica subentry, as determined by the **ibm-replicaServerID** attribute, is a master, peer, or gateway server if there are any replication agreement entries for the replication context under this replica subentry. If set to false, the server that is represented by this replica subentry, as determined by the **ibm-replicaServerID** attribute, is a forwarding server if there are any replication agreement entries for the replication context under this replica subentry. Example: `ibm-replicationServerIsMaster: true`

For the examples in Table 69 on page 497, the replica subentry represents a supplier server with a server ID of `supplier1`. It is also the master server under the replication context (`o=ibm`) where this replica subentry exists.

Replication agreements

A replication agreement is an entry in the directory with the structural object class **ibm-replicationAgreement** created directly under a replica subentry to define replication from the server represented by the subentry to another server. See Table 70 for the required attributes for the **ibm-replicationAgreement** objectclass. See Table 71 on page 499 for the optional attributes for the **ibm-replicationAgreement** objectclass. A replication agreement entry is like a **replicaObject** entry used in basic replication. This object represents an individual connection from a supplier server to a consumer server. A replica subentry might have any number of replication agreement entries defined under it to specify each supplier agreement this server has under this replication topology.

Table 70. **ibm-replicationAgreement** *objectclass schema definition (required attributes)*

Attribute description and example
cn Common name of the replication agreement entry. This attribute does not affect advanced replication configuration. Example: `cn: agreement1`
ibm-replicaConsumerID Identifies the server ID of the consumer server. This value matches the **ibm-serverID** attribute value in the root DSE entry of the consumer server or a warning message is issued in the LDAP server log when the replication agreement initializes. This attribute is case-sensitive and must exactly match the **ibm-serverID** attribute value on the consumer server. Example: `ibm-replicaConsumerID: consumer1`
ibm-replicaCredentialsDN Specifies the distinguished name (DN) of the entry containing the credentials entry used to authenticate to the consumer server. See “Credentials entries” on page 500 for more information about credential entries. Example: `ibm-replicaCredentialsDN: cn=consumer1,cn=localhost`

Table 70. **ibm-replicationAgreement** *objectclass schema definition (required attributes) (continued)*

Attribute description and example
ibm-replicaURL Specifies the LDAP URL of the consumer server. The LDAP URL syntax is fully documented in RFC 2255: *The LDAP URL Format*. The prefix of the LDAP URL indicates whether a non-secure or secure connection is used between the supplier and consumer servers. If the LDAP URL prefix is `ldap://`, a non-secure connection is used. If the LDAP URL prefix is `ldaps://`, a secure connection is used. You should use a secure connection when replicating so that sensitive data, such as **userPassword** values, are not exposed in the clear. See "SSL/TLS and advanced replication" on page 539 for more information about using SSL/TLS in an advanced replication environment. Example: `ibm-replicaURL: ldaps://consumer1.ibm.com:500`

For the examples in Table 70 on page 498, when the replicating server receives and successfully finishes an update request, the update is also sent to the consumer server with an ID of `consumer1` that is on host name `consumer1.ibm.com` on secure port `500`. The replicating server performs a simple or SASL EXTERNAL bind operation using the information provided in the credentials entry `cn=consumer1,cn=localhost`.

Table 71. **ibm-replicationAgreement** *objectclass schema definition (optional attributes)*

Attribute description and example
description Provides an additional text field for extra information pertaining to the replication agreement entry. This attribute does not affect advanced replication configuration. Example: `description: Represents the replication agreement from the supplier1 server to the consumer1 server`
ibm-replicaScheduleDN Specifies the DN of a schedule entry that determines when replication updates are sent to this consumer. If a schedule DN is not specified, advanced replication defaults to "immediate" replication mode. See "Schedule entries" on page 503 for more information about advanced replication scheduling. Example: `ibm-replicaScheduleDN: cn=schedule,o=ibm`
ibm-replicationCreateMissingEntries A boolean (true or false) indicating whether missing parent entries are to be created on the consumer server. If set to true, the missing parent entries are automatically created by the supplier server and replicated to the consumer server. If set to false or if the attribute is not specified, the missing parent entries are not created on the consumer server. Example: `ibm-replicationCreateMissingEntries: true`

Table 71. **ibm-replicationAgreement** *objectclass schema definition (optional attributes) (continued)*

Attribute description and example
ibm-replicationExcludedCapability A multi-valued attribute that lists the OIDs of features that the consumer server does not support. Operations related to these capabilities are excluded from the updates sent to the consumer in this replication agreement. If this attribute is not specified, no capabilities are excluded from being replicated. Only the following capabilities are allowed to be excluded: • 1.3.18.0.2.32.4 – IBM filtered ACLs (only supported on non-z/OS IBM Tivoli Directory Servers) • 1.3.18.0.2.32.98 – z/OS IBM Tivoli Directory Server ACL filters Example: `ibm-replicationExcludedCapability: 1.3.18.0.2.32.4`
ibm-replicationFilterDN Specifies the DN of a replication filter entry that contains filters that include or exclude the replication of certain entries or attribute types to the consumer server. See “Partial replication” on page 536 for additional information about using partial replication. Example: `ibm-replicationFilterDN: cn=filter,o=ibm`
ibm-replicationOnHold A boolean (true or false) indicating whether advanced replication from the replication agreement is suspended or not. If set to true, replication updates from the supplier server to the consumer server are queued until this attribute value is set to false. If set to false or this attribute is not specified, replication updates are handled normally. This attribute value is also modified by the **Cascading control replication** and the **Control replication** extended operations. See “Cascading control replication” on page 696 for more information about the **Cascading control replication** extended operation. See “Control replication” on page 700 for more information about the **Control replication** extended operation. Example: `ibm-replicationOnHold: false`

To aid in enforcing the accuracy of the data within the replication agreement entry, when the supplier binds to the consumer, it retrieves the **ibm-serverID** attribute from the root DSE entry and compares it to the **ibm-replicaConsumerID** attribute value. A warning is logged in the LDAP servers job log if these server IDs do not match.

You can designate that part of a replicated subtree not replicate by adding the **ibm-replicationContext** auxiliary class to the root of the subtree, without defining any replica subentries.

Credentials entries

Because the replication agreement entry can be replicated, a DN to credentials object is used in the **ibm-replicaCredentialsDN** attribute value. This allows the supplier server credentials entry to be stored in an area of the DIT that is not replicated. Replicating the supplier server credentials entries (where 'clear text' passwords must be obtainable) represents a potential security exposure. The **cn=localhost** suffix in an LDBM or TDBM backend is an appropriate location for the creation of supplier server credential entries. **cn=localhost** is a special suffix

that is exempt from replication. Also, an additional option is needed when using the **ds2ldif** utility to unload its descendant entries. See "ds2ldif utility" on page 225 for more information.

The objectclass of the entry that is specified in the **ibm-replicaCredentialsDN** attribute value in the replication agreement indicates that the authentication method used by the supplier to authenticate with the consumer server. If the entrys objectclass is **ibm-replicationCredentialsSimple**, the supplier server uses a simple bind to authenticate to the consumer. See Table 72 for the required attributes of the **ibm-replicationCredentialsSimple** objectclass. If the entrys objectclass is **ibm-replicationCredentialsExternal**, the supplier server performs a SASL EXTERNAL bind to the consumer server. See Table 73 on page 502 for the optional attributes of the **ibm-replicationCredentialsExternal** objectclass.

A consumer server credentials entry is required on the consumer server to identify the distinguished name that the supplier server is using to perform a simple or SASL EXTERNAL bind. See "Consumer server entries" on page 514 for more information about the consumer server credential entries.

Table 72. **ibm-replicationCredentialsSimple** *objectclass schema definition (required attributes)*

Attribute description and example
replicaBindDN Specifies the LDAP distinguished name that the replicating server uses to bind with the consumer server when sending directory updates. Example: `replicaBindDN: cn=supplier,cn=localhost`
replicaCredentials Contains the authentication information needed for the replicating server to authenticate with the consumer server using the distinguished name specified in the **replicaBindDN** attribute value. This password value is encrypted if it is added or modified when the **secretEncryption** configuration option is set to **AES** or **DES** under the backend containing the replication agreement. If **secretEncryption** is set to **AES** or **DES**, directory security improves because the password is no longer stored in the directory in clear text. Example: `replicaCredentials: secret`

For the examples in Table 72, the replication agreement uses a simple bind to the consumer server using a bind DN of `cn=supplier,cn=localhost` and a bind password of `secret`.

Table 73. **ibm-replicationCredentialsExternal** *objectclass schema definition (optional attributes)*

Attribute description and example
ibm-replicaKeyFile Specifies the path and file name of the SSL/TLS key database file, the name of the RACF key ring, or the name of the PKCS #11 token to be used by the replication agreement to perform an SASL EXTERNAL bind. Specifying a value here, overrides the default that comes from the **sslKeyRingFile** configuration option. See "sslKeyRingFile name" on page 132 for the acceptable formats for this attribute value. **Note:** If a value is specified for this attribute, it must be the same for all replication agreements in the server. The LDAP server only supports having one opened SSL/TLS key database file, RACF key ring, or PKCS #11 token at a time. It is suggested that all SSL certificates that must be used by the server be placed in one SSL/TLS key database file, RACF key ring, or PKCS #11 token. Example: `ibm-replicaKeyFile: /home/server1/server1.kdb`
ibm-replicaKeyLabel Specifies the label of the certificate that is used for LDAP server-client authentication for the SASL EXTERNAL bind. The certificate label must exist in the SSL/TLS key database file, RACF key ring, or PKCS #11 token being used for this credentials entry, as specified by the **ibm-replicaKeyFile** attribute value or the **sslKeyRingFile** configuration option if **ibm-replicaKeyFile** is not specified. Specifying a value here, overrides the default that comes from the **sslCertificate** configuration option. If the **sslCertificate** configuration option is not specified or is set to **none**, the default SSL certificate in the **ibm-replicaKeyFile** attribute value (or the **sslKeyRingFile** configuration option if **ibm-replicaKeyFile** is not specified) is used. See "sslCertificate {certificateLabel | none}" on page 130 for more about the **sslCertificate** option. Example: `ibm-replicaKeyLabel: EXTERNAL1`

ibm-replicaKeyPwd
 	
Specifies the password protecting access to the SSL/TLS key database file. It can also be used to specify a fully qualified file name where the password for the SSL/TLS key database file is stashed. Only specify this attribute if the **ibm-replicaKeyFile** attribute value (or the **sslKeyRingFile** configuration option if **ibm-replicaKeyFile** is not specified) is an SSL/TLS key database file.
 	
If using an SSL stash file, it must be specified in the following format:
`file://filename`
 	
where `filename` is the fully qualified z/OS z/OS UNIX System Services file system location of the SSL stash file.
 	
This password value is encrypted if it is added or modified when the **secretEncryption** configuration option is set to **AES** or **DES** under the backend containing the replication agreement. If **secretEncryption** is set to **AES** or **DES**, directory security improves because the password is no longer stored in the directory in clear text.
 	
Example:
`ibm-replicaKeyPwd: secret`

For the examples in Table 73, the replication agreement uses a SASL EXTERNAL bind to the consumer server with the SSL certificate label `EXTERNAL1` in SSL key database file `/home/server1/server1.kdb` that has a password of `secret`.

A SASL EXTERNAL bind requires a secure connection from the replicating server to the replica server. The replication agreement entry must use an **ibm-replicaURL** attribute value with an LDAP URL prefix of `ldaps://` to signify an SSL connection. The replicating server must have read access to the SSL/TLS key database file, RACF key ring, or PKCS #11 token that is specified in the **sslKeyRingFile**

configuration option or the **ibm-replicaKeyFile** attribute value in the SASL EXTERNAL supplier server credentials entry. If the optional attribute values in Table 73 on page 502 are not specified in the SASL EXTERNAL supplier server credentials entry, the default SSL configuration in the LDAP server configuration file is used.

See "SSL/TLS and advanced replication" on page 539 for more information about using SSL/TLS in an advanced replication environment.

Schedule entries

An LDAP administrator can schedule advanced replication to occur at optimal times when network traffic is minimal for each individual replication agreement. Each replication agreement entry is allowed to have an **ibm-replicaScheduleDN** attribute value optionally specified. This attribute value identifies the distinguished name (DN) of a weekly schedule entry, that has an object class of **ibm-replicationWeeklySchedule**. See Table 74 for the schema definition of the **ibm-replicationWeeklySchedule** objectclass. The weekly schedule entry allows an LDAP administrator to specify the distinguished name (DN) of additional entries that point to one or more daily replication schedule entries. If the distinguished name in the **ibm-replicaScheduleDN** attribute value cannot be found or is not a weekly schedule entry, advanced replication continues by ignoring the weekly replication schedule.

Table 74. **ibm-replicationWeeklySchedule** *objectclass schema definition (optional attributes)*

Attribute description and example
cn Common name of the weekly replication schedule entry. This attribute does not affect advanced replication configuration. Example: `cn: myweekly`
description Provides an additional text field for extra information pertaining to the weekly schedule entry. This attribute does not affect advanced replication configuration. Example: `description: Weekly schedule for advanced replication`
ibm-replWeeklySchedName Descriptive name for the weekly schedule entry. This attribute does not affect advanced replication configuration. Example: `ibm-replWeeklySchedName: Weekly schedule for agreement 1`
ibm-scheduleMonday Specifies the distinguished name (DN) of a daily replication schedule entry for Monday. Example: `ibm-scheduleMonday: cn=monday,o=ibm`

Table 74. **ibm-replicationWeeklySchedule** *objectclass schema definition (optional attributes) (continued)*

Attribute description and example
ibm-scheduleTuesday Specifies the distinguished name (DN) of a daily replication schedule entry for Tuesday. Example: `ibm-scheduleTuesday: cn=tuesday,o=ibm`
ibm-scheduleWednesday Specifies the distinguished name (DN) of a daily replication schedule entry for Wednesday. Example: `ibm-scheduleWednesday: cn=wednesday,o=ibm`
ibm-scheduleThursday Specifies the distinguished name (DN) of a daily replication schedule entry for Thursday. Example: `ibm-scheduleThursday: cn=thursday,o=ibm`
ibm-scheduleFriday Specifies the distinguished name (DN) of a daily replication schedule entry for Friday. Example: `ibm-scheduleFriday: cn=friday,o=ibm`
ibm-scheduleSaturday Specifies the distinguished name (DN) of a daily replication schedule entry for Saturday. Example: `ibm-scheduleSaturday: cn=saturday,o=ibm`
ibm-scheduleSunday Specifies the distinguished name (DN) of a daily replication schedule entry for Sunday. Example: `ibm-scheduleSunday: cn=sunday,o=ibm`

A daily replication schedule entry has an object class of **ibm-replicationDailySchedule**. See Table 75 on page 505 for the schema definition of the **ibm-replicationDailySchedule** objectclass. A daily replication schedule entry allows an LDAP administrator to accomplish the following replication scheduling:

- Configure the time each day to start advanced replication for that replication agreement. This is accomplished by using the multi-valued **ibm-replicationImmediateStart** attribute.
- Allows replication to be turned off by specifying a batch time. This drains the replication queue and replication waits until the next scheduled time once the queue is fully drained. This is done by using the multi-valued **ibm-replicationBatchStart** attribute.
- Advanced replication can be turned on and off multiple times each day.

Table 75. **ibm-replicationDailySchedule** *objectclass schema definition (optional attributes)*

Attribute description and example
cn Common name of the daily schedule entry. This attribute does not affect advanced replication configuration. Example: `cn: mydaily`
description Provides an additional text field for extra information pertaining to the daily schedule entry. This attribute does not affect advanced replication configuration. Example: `description: Each day stops replication at 6:30 a.m. and then restarts at 9:45 a.m. and continues for the rest of the day until 6:30 a.m. the following day.`
ibm-replDailySchedName Descriptive name for the daily schedule entry. This attribute does not affect advanced replication configuration. Example: `ibm-replDailySchedName: Daily schedule for replication agreement 1`
ibm-replicationBatchStart A multi-valued attribute that indicates the time batch replication starts. All updates in the replication queue are replicated at the time specified and then replication waits until the next scheduled time. **Note:** If advanced replication is waiting, it is not allowed to be resumed with either a **Cascading control replication** extended operation or a **Control replication** extended operation. However, "replicate now" on a **Control replication** extended operation can be used to immediately drain the replication queue if replication is waiting. See “Cascading control replication” on page 696 for more information about the **Cascading control replication** extended operation. See “Control replication” on page 700 for more information about the **Control replication** extended operation. The attribute value format is: `Thhmmss` where: `hh` - Hour based on a 24 hour clock (00 – 23) `mm` - Minutes (00-59) `ss` - Seconds (00-59) Example: `ibm-replicationBatchStart: T063000` This value indicates that queued replication updates are replicated at 6:30 AM until the replication queue is drained; then replication waits.

Table 75. **ibm-replicationDailySchedule** *objectclass schema definition (optional attributes) (continued)*

Attribute description and example
ibm-replicationImmediateStart A multi-valued attribute that indicates when advanced replication immediately starts and continues until the next **ibm-replicationBatchStart** attribute value or replication is otherwise suspended. Advanced replication is allowed to be suspended or resumed with a **Cascading control replication** extended operation or a **Control replication** extended operation. See "Cascading control replication" on page 696 for more information about the **Cascading control replication** extended operation. See "Control replication" on page 700 for more information about the **Control replication** extended operation. The attribute value format is: `Thhmmss` where: `hh` - Hour based on a 24 hour clock (00 – 23) `mm` - Minutes (00-59) `ss` - Seconds (00-59) Example: `ibm-replicationImmediateStart: T094500` This value indicates that replication starts at 9:45 a.m..
ibm-replicationTimesUTC A boolean (true or false) indicating whether the time values specified in the **ibm-replicationBatchStart** and **ibm-replicationImmediateStart** attributes are in Coordinated Universal Time or local time. If set to true, Coordinated Universal Time is used for time values. If set to false, or the attribute is not specified, local time is used for time values. Example: `ibm-replicationTimesUTC: true`

Assuming that each daily schedule in the weekly schedule entry uses the examples in Table 75 on page 505, advanced replication occurs daily as follows:

- Because an **ibm-replicationBatchStart** attribute value of `T06300` is specified, the replication queue is drained and replication waits at 6:30 a.m. each day. All future replication updates are queued.
- At 9:30 a.m. each day, advanced replication restarts because an **ibm-replicationImmediateStart** attribute value is specified. Replication immediately starts and continues until the next day at 6:30 a.m..

When the LDAP server starts and there is a weekly replication schedule entry configured, advanced replication inherits the state of the most recent **ibm-replicationBatchStart** or **ibm-replicationImmediate** time. If the weekly schedule examples are used in Table 75 on page 505 and the LDAP server starts at 7:00 a.m., replication is suspended until 9:30 a.m. when the next **ibm-replicationImmediate** time is encountered. This processing occurs even if there is a missing daily schedule in the weekly schedule entry.

Consumer server entries

If the consumer server is a read only replica server, the only required replication-related entry is the consumer server credentials entry. If the consumer server is a peer or forwarding server, a replica subentry and a consumer server credentials entry are required. The consumer server credentials entry must exist under the **cn=configuration** suffix in the CDBM backend.

Note: The consumer server credentials entry differs from the supplier server credentials entry. See "Credentials entries" on page 500 for more information about the supplier server credentials entry.

The consumer server credentials entry is used on the consumer server to verify that it is actually a supplier server performing a simple or SASL EXTERNAL bind. A consumer server only accepts update operations from its supplier server and any LDAP administrator when using the **Server Administration** control. There are two types of consumer server credential entries that can be used, one that has an objectclass of **ibm-slapdReplication** and the other has an objectclass of **ibm-slapdSupplier**.

When a supplier server replicates updates to its consumer server, a special entry is used to indicate that the supplier server has master level access to the consumer server. Master level access bypasses ACL and entry owner restrictions and allows updates to be made even when the server is a read-only consumer, cascading consumer, or under a quiesced replication context. If the supplier server authenticates to the consumer server with a simple bind, the DN specified by the **replicaBindDN** attribute value in the replication agreement entry is used as the bind DN. If the supplier server authenticates to the consumer server with a SASL EXTERNAL bind, the bind DN is extracted from the SSL certificate unless the **sslMapCertificate** configuration options first value is set to **replace**. See "sslMapCertificate {off | check | add | replace} {fail | ignore}" on page 133 for more information about certificate mapping.

If the **ibm-slapdMasterDN** attribute value in an **ibm-slapdReplication** entry matches the bind DN, the supplier server (or user) is allowed master level access to all replication contexts. If the **ibm-slapdMasterDN** attribute value in an **ibm-slapdSupplier** entry matches the bind DN, the supplier server (or user) is only allowed master level access to the replication contexts indicated by the multi-valued **ibm-replicaSubtree** attribute value.

Note: The consumer server credentials entry must be present on both the consumer and supplier servers and exist under the **cn=configuration** suffix in the CDBM backend. The topology entries are the only way for the servers to know their roles in the topology as a whole, therefore, are needed on all the servers in the topology.

Table 76. **ibm-slapdReplication** *objectclass schema definition (required and optional attributes)*

Attribute description and example
cn A required attribute that specifies the common name of the consumer server credentials entry. Example: `cn: master server`

Table 76. **ibm-slapdReplication** *objectclass schema definition (required and optional attributes) (continued)*

Attribute description and example
ibm-slapdMasterDN Specifies the distinguished name (DN) that the supplier server uses to authenticate with the consumer server. If the supplier server authenticates to the consumer server with a simple bind, this value matches the **replicaBindDN** attribute value in the simple bind supplier server credentials entry used by the replication agreement entry. If the supplier server authenticates to the consumer server with a SASL EXTERNAL bind, this value matches the bind DN extracted from the SSL certificate unless the **sslMapCertificate** configuration options first value is set to **replace**. See "sslMapCertificate {off | check | add | replace} {fail | ignore}" on page 133 for more information about certificate mapping. Example: `ibm-slapdMasterDN: cn=supplier,cn=localhost`
ibm-slapdMasterPW Contains the simple bind authentication information needed for the replicating server to authenticate with the consumer server using the **ibm-slapdMasterDN**. This password value matches the **replicaCredentials** attribute value in the simple bind supplier server credentials entry used by the replication agreement entry. This password value is encrypted if it is added or modified when the **secretEncryption** configuration option is set to **AES** or **DES** in the CDBM backend. If **secretEncryption** is set to **AES** or **DES**, directory security improves because the password is no longer stored in the directory in clear text. If a SASL EXTERNAL bind is used, this attribute value is not specified. **Note:** This value is only used if the entry specified in the **ibm-slapdMasterDN** attribute value does not exist under a configured suffix in the LDAP server. Example: `ibm-slapdMasterPW: secret`
ibm-slapdMasterReferral A single valued attribute that contains the LDAP URL of the supplier server. The LDAP URL syntax is documented in RFC 2255: *The LDAP URL Format*. If an update operation is done by a user other than the supplier server or any LDAP administrator with the **Server Administration** control, this value is returned as one of the referral values. See "Replication topology hints and tips" on page 533 for more information about referrals with advanced replication. Example: `ibm-slapdMasterReferral: ldap://master1.ibm.com:500`
ibm-slapdNoReplConflictResolution A boolean (true or false) indicating whether the consumer server participates in replication conflict resolution. If set to true, the consumer server does not participate in conflict resolution. If set to false, or the attribute is not specified, the consumer server does participate in conflict resolution. Conflict resolution is used to automatically attempt to resolve conflicts with entries that are no longer synchronized between a supplier and consumer server. The **modifyTimestamp** attribute value of the entry is used to detect a conflict between the two servers. Example: `ibm-slapdNoReplConflictResolution: true`

For the examples in Table 76 on page 507, the supplier server located at `master1.ibm.com` on non-secure port `500` does a simple bind to the consumer server by binding with the `cn=supplier,cn=localhost` entry and specifying a password of `secret`. The consumer server is not configured for conflict resolution.

Table 77. **ibm-slapdSupplier** *objectclass schema definition (required and optional attributes)*

Attribute description and example
cn A required attribute that specifies the common name of the consumer server credentials entry. Example: `cn: master server`
ibm-slapdMasterDN Specifies the distinguished name (DN) that the supplier server uses to authenticate with the consumer server. If the supplier server authenticates to the consumer server with a simple bind, this value matches the **replicaBindDN** attribute value in the simple bind supplier server credentials entry used by the replication agreement entry. If the supplier server authenticates to the consumer server with a SASL EXTERNAL bind, this value matches the bind DN extracted from the SSL certificate unless the **sslMapCertificate** configuration options first value is set to **replace**. See "sslMapCertificate {off | check | add | replace} {fail | ignore}" on page 133 for more information about certificate mapping. Example: `ibm-slapdMasterDN: cn=supplier,cn=localhost`
ibm-slapdMasterPW Contains the simple bind authentication information needed for the replicating server to authenticate with the consumer server using the **ibm-slapdMasterDN**. This password value matches the **replicaCredentials** attribute value in the simple bind supplier server credentials entry used by the replication agreement entry. This password value is encrypted if it is added or modified when the **secretEncryption** configuration option is set to **AES** or **DES** in the CDBM backend. If **secretEncryption** is set to **AES** or **DES**, directory security improves because the password is no longer stored in the directory in clear text. If a SASL EXTERNAL bind is used, this attribute value is not specified. **Note:** This value is only used if the entry specified in the **ibm-slapdMasterDN** attribute value does not exist under a configured suffix in the LDAP server. Example: `ibm-slapdMasterPW: secret`
ibm-slapdReplicaSubtree A multi-valued attribute that specifies the distinguished names of replication contexts that are subject to this consumer server credentials entry. The bound user has master server level access to the replication contexts that are specified for this attribute. Example: `ibm-slapdReplicaSubtree: o=ibm`

For the examples in Table 77, when the supplier server replicates updates to the `o=ibm` replication context on the consumer server, the supplier server performs a simple bind using the `cn=supplier,cn=localhost` entry and specifying a password of `secret`.

Things to consider before configuring advanced replication

Before setting up an advanced replication configuration, there are some administrative responsibilities that must be considered. In order to ensure that replication is operating smoothly and that your replicas are staying up-to-date, an administrator must take some periodic actions to monitor the replication status. After advanced replication is correctly configured, it continues to automatically propagate updates to all defined replica servers. However, if errors occur, human intervention might be required to fully correct the problem.

Detailed status and error information is available to an LDAP administrator by querying the operational attributes in the replication agreement entries. See "Monitoring and diagnosing advanced replication problems" on page 543 for a description of the information available. Configuring multiple master servers adds to the potential error cases that an LDAP administrator must be aware of. If the same entry is updated at two different master servers at approximately the same time, those updates are likely to conflict when they are replicated to other servers in the advanced replication topology. The advanced replication conflict resolution support is designed to detect and resolve conflicts that might occur. See "Replication conflict resolution" on page 493 for more information about replication conflict resolution.

Consider the following when planning an advanced replication environment:

1. Determine if an existing Directory Information Tree (DIT) subtree is to be introduced into a replication topology or if a new subtree is to be added to the server after the replication topology is established. It is suggested that all servers that serve as a supplier server are put into maintenance mode until the replication topology entries are loaded on all servers. This ensures that external updates to the subtree are not lost while configuring advanced replication. See "Advanced replication maintenance mode" on page 536 for more information.
   a. If using an existing subtree for advanced replication:
      1) Modify the subtree to add the auxiliary objectclass **ibm-replicationContext**. If the **ibm-replicationContext** auxiliary objectclass is added to a non-suffix level entry in the directory, explicit **aclEntry** and **entryOwner** attribute values are required. The **entryOwner** attribute value must start with an *access-id:*.
      2) Unload the entire subtree to an LDIF file by using the **ds2ldif** utility. See "ds2ldif utility" on page 225 for additional information about the **ds2ldif** utility
      3) For each server participating in the replication topology, add the unloaded entries to the server by using the **ldapadd** or **ldif2ds** utilities. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information about the **ldapadd** utility. See "ldif2ds utility" on page 235 for additional information about the **ldif2ds** utility.
   b. If using a new subtree for advanced replication, verify that the subtree has an auxiliary objectclass of **ibm-replicationContext**. For each server participating in the replication topology, add the same entries to all servers by using the **ldapadd** or **ldif2ds** utilities. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information about the **ldapadd** utility. See "ldif2ds utility" on page 235 for additional information about the **ldif2ds** utility.
2. For each consumer server in the replication topology, load master bind and referral information under the **cn=configuration** suffix in the CDBM backend. Consumer server credential entries with an objectclass of **ibm-slapdSupplier** or

ibm-slapdReplication must be added. If using a consumer server credentials entry with an objectclass of **ibm-slapdSupplier**, the replication context must be added to the **ibm-slapdReplicaSubtree** attribute value. See "Consumer server entries" on page 514 for more information.

3. For each supplier server in the replication topology, supplier server credential entries must be added for each unique consumer server in the replication context. A supplier server credential entry enables the supplier server to authenticate with the consumer server by using a simple or SASL EXTERNAL bind. If the objectclass of the supplier server credentials entry is **ibm-replicationCredentialsSimple**, a simple bind is used. If the objectclass of the supplier server credentials entry is **ibm-replicationCredentialsExternal**, a SASL EXTERNAL bind is used.

Note: There are no requirements for placing supplier server credential entries within a specific subtree. If supplier server credential entries are not replicated, use the **cn=localhost** subtree that does not allow the replication of entries. If supplier server credential entries are replicated outside the scope of the replication context being configured, consider using the **cn=ibmpolicies** subtree. When the **cn=ibmpolicies** subtree is configured for advanced replication, schema modifications are also replicated. See "Replication of schema and password policy updates" on page 534 for more information about schema replication.

The following steps are used to deploy the replication topology on all servers:

1. On a supplier server in the topology, use the **ldapadd** utility with the **Server Administration** and the **Do Not Replicate** controls to add the replication topology entries. The replication topology entries are the following:
   a. Replication context with an objectclass of **ibm-replicationContext**. See "Replication contexts" on page 496 for more information.
   b. Replica group with an objectclass of **ibm-replicaGroup**. See "Replica groups" on page 496 for more information.
   c. Replica subentry with an objectclass of **ibm-replicaSubentry**. See "Replica subentries" on page 497 for more information.
   d. Replication agreement with an objectclass of **ibm-replicationAgreement**. See "Replication agreements" on page 498 for more information.
2. On the replication context added in the previous step, use the **Replication topology** extended operation in the **ldapexop** utility. This synchronizes all replication topology entries for each consumer server in the replication context. See "ldapexop utility" on page 255 for more information about the **ldapexop** utility.

When these steps are complete, the supplier servers are moved out of maintenance mode.

Advanced replication configuration examples

This topic provides examples of the different replication topologies that can be configured. It provides example LDIF data that includes the host names, IP addresses, ports, server IDs, and passwords.

Suppliers and consumers

In advanced replication, updates are propagated from one LDAP server to another through a replication queue. The server that enters updates into the replication queue is called a supplier. The server that absorbs these changes is called the consumer. The queue is maintained on the supplier.

- **Host names and ports:** Provide the supplier with enough information to connect to the consumer.
- **Server IDs:** Strings that enable one LDAP server to identify other LDAP servers in the topology.
- **Bind DNs and passwords:** The supplier connects to the consumer using the LDAP protocol. In LDAP terminology, this is called a bind. The bind requires a bind Distinguished Name (DN) and a password.

The following examples demonstrate setup for a topology consisting of a maximum of three servers.

Note: The host name of the consumer resolves correctly from the supplier. If not, the supplier cannot connect to the consumer and advanced replication fails.

Table 78. Topology setup

Server	Host name
Server 1	server1.us.ibm.com
Server 2	server2.us.ibm.com
Server 3	server3.us.ibm.com

Each LDAP server in Table 78 is listening on port 389, which is the default LDAP port.

These examples assume that a simple bind is being done from the supplier server to the consumer server. Each example assumes the following bind DN and password for all supplier-consumer agreements.

- **DN:** `cn=bindtoconsumer`
- **Password:** `iamsupplier`

Server ID

In the examples, the server ID of each server is the role of that server in the topology. That is, in the Master-Replica topology, the master identifies the server ID as `Master` and the replica is identified as `Replica`. In the Peer-to-Peer topology, one peer is `Peer1` and the other is `Peer2`. In the Master-Forwarder-Replica topology, the master is `Master`, the forwarder is `Forwarder`, and the replica is `Replica`. In the Gateway topology, the gateway servers are `Gateway1` and the other is `Gateway2` and the replica is `Replica`.

Do not change the server ID for a server. When the z/OS LDAP server is first configured with a CDBM backend, the server ID is generated as an IBM entry UUID value in the **ibm-slapdServerID** attribute value in the **cn=configuration** entry. For convenience, the server ID is published in the rootDSE entrys attribute **ibm-serverID**. The server ID is only allowed to be modified when there are no replica subentries defined in the server. See Table 10 on page 139 for more information about the **ibm-slapdServerID** attribute value.

Advanced replication related entries summary

For convenience, this section quickly summarizes the various types of entries that are used to build an advanced replication topology.

Supplier server entries

- **Replication context:** This is the root entry for the subtree that is to be replicated. It must have an auxiliary objectclass of **ibm-replicationContext.** To replicate a subtree o=ibm,c=us, the replication context might be:

  ```
  dn: o=ibm,c=us
  objectclass: top
  objectclass: organization
  objectclass: ibm-replicationContext
  o: ibm
  ```

  All other replication entries except for the credential and schedule entries must be under the replication context. The credential and schedule entries can be anywhere in the DIT.

- **Replica group:** This entry is not important apart from the fact that all the advanced replication-related entries exist under this entry. It must have the **ibm-replicaGroup** objectclass. For example:

  ```
  dn: ibm-replicaGroup=default, o=ibm,c=us
  objectclass: top
  objectclass: ibm-replicaGroup
  ibm-replicaGroup: default
  ```

- **Replica subentry:** These types of entries declare the servers that are taking part in the advanced replication topology. Each server participating in the topology has one subentry. If a server is represented by more than one subentry under a replication context, unexpected behavior might result. This is done by having more than one subentry under a replication context containing the same **ibm-replicaServerID** attribute value. This entry has the **ibm-replicaSubentry** objectclass. For example:

  ```
  dn: ibm-replicaServerId=Peer1,ibm-replicaGroup=default, o=ibm,c=us
  objectclass: top
  objectclass: ibm-replicaSubentry
  ibm-replicaServerId: Peer1
  ibm-replicationServerIsMaster: true
  cn: Peer1
  description: Peer1
  ```

  As shown in the replica subentry example, the entry has the server ID of the participating server, Peer1. It has an attribute called **ibm-replicationServerIsMaster**. When this attribute is set to true, the server is a read/write copy.

- **Replication agreements:** These types of entries occur under replica subentries. When these entries appear under a specific server's replica subentry, they define a replication agreement from that server to some other server in the topology. For example:

  ```
  dn: cn=Peer2, ibm-replicaServerId=Peer1,ibm-replicaGroup=default,o=ibm,c=us
  objectclass: top
  objectclass: ibm-replicationAgreement
  cn: Peer2
  ibm-replicaConsumerId: Peer2
  ibm-replicaUrl: ldap://server2.us.ibm.com:389
  ibm-replicaCredentialsDN: cn=Peer1BindCredentials, cn=localhost
  description: Replication agreement from Peer1 to Peer2
  ```

  The replication agreement example is from Peer1 to Peer2. The supplier is Peer1 as the agreement occurs under the subentry for Peer1. The consumer is Peer2. The server Peer2 is on server2.us.ibm.com and is listening on port 389. Peer1 binds to Peer2 using the credentials defined in the entry (cn=Peer1BindCredentials,cn=localhost).

- **Replication credentials:** If a simple bind is used by the supplier server to authenticate with the consumer server, this entry defines the bind DN and password that is used. This credential entry uses the **ibm-replicationCredentialsSimple** objectclass. If a SASL EXTERNAL bind is used by the supplier server to authenticate with the consumer server, see "Credentials entries" on page 500 for information about the **ibm-replicationCredentialsExternal** objectclass. For example:

  ```
  dn: cn=Peer1BindCredentials, cn=localhost
  objectclass: ibm-replicationCredentialsSimple
  cn: ReplicaBindCredentials
  replicaBindDN: cn=bindtoconsumer
  replicaCredentials: iamsupplier
  description: Bind Credentials on Peer1 to be used to bind to other servers.
  ```

  The replication credential example defines the **replicaBindDN** as `cn=bindtoconsumer` and the password as `iamsupplier`. Take note of the description. The same credentials entry can be used for multiple replication agreements.

Consumer server entries

If the consumer server is a read only replica server, the only required replication-related entry is the consumer server credentials entry. If the consumer server is a peer or forwarding server, a replica subentry and a consumer server credentials entry are required. The consumer server credentials entry identifies the distinguished name and optionally the password value that the supplier server uses to authenticate with the consumer server. There are two types of credential entries that can be used on the consumer.

Type 1 example:

```
dn: cn=Master server,cn=configuration
objectclass: ibm-slapdReplication
cn: master server
ibm-slapdMasterDN: cn=bindtoconsumer
ibm-slapdMasterPW: iamsupplier
ibm-slapdMasterReferral: ldap://localhost:1389
```

Type 2 example:

```
dn: cn=Supplier s1,cn=configuration
objectclass: ibm-slapdSupplier
cn: Supplier s1
ibm-slapdMasterDN: cn=bindtoconsumer
ibm-slapdMasterPW: iamsupplier
ibm-slapdReplicaSubtree: o=ibm,c=us
```

The use of the credential established by type 2 is limited to the **ibm-slapdReplicaSubtree** only. Therefore, suppliers binding with bind DN as `cn=bindtoconsumer` and password as `iamsupplier` supplies only to the `o=ibm,c=us` subtree, unless another credential entry gives rights to another subtree. The type 1 credential entry is global across the LDAP server. When a type 2 entry is defined in the **cn=configuration** suffix in the CDBM backend, any subtree can be supplied to if a supplier authenticates with bind DN of `cn=bindtoconsumer` and password of `iamsupplier`.

Note: The consumer server credentials entry must be present on both the consumer and supplier servers and exist under the **cn=configuration** suffix in the CDBM backend. The topology entries are the only way for the servers to know their roles in the topology as a whole, therefore, are needed on all the servers in the topology.

Creating a master-replica topology

This example describes deploying the most basic of all the topologies, the master-replica topology. It has one read/write server and one read-only server.

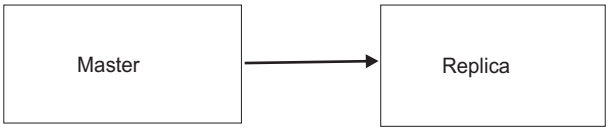


Figure 49. Master-replica topology

- The first step when building a topology is to define:
  1. **Replication context:** `o=ibm,c=us`
  2. **Supplier(s):** LDAP server on `server1.us.ibm.com:389` is the only supplier. The server ID is `Master`. It supplies updates to the LDAP server on `server2.us.ibm.com:389`.
  3. **Consumer(s):** LDAP server on `server2.us.ibm.com:389` is the only consumer. The server ID is `Replica`. It consumes updates from the LDAP server on `server1.us.ibm.com:389`.
  4. **read/write server(s):** LDAP server on `server1.us.ibm.com:389`, with ID `Master` is the only read/write server.
  5. **Read-only server(s):** LDAP server on `server2.us.ibm.com:389`, with ID `Replica` is the only read-only server.

  Configuration changes: Because of these examples, some CDBM changes are needed for the master and the replica server for replication to work correctly.

  Note: These are done here for this example ONLY. The server ID is only allowed to be modified when there are no replica subentries defined in the server. See Table 10 on page 139 for more information about the **ibm-slapdServerID** attribute value.

  1. **Server IDs:**
     - On the master server, apply the following modify using the **ldapmodify** utility. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for additional information about the **ldapmodify** utility.

       ```
       dn: cn=configuration
       changetype: modify
       replace: ibm-slapdserverid
       ibm-slapdserverid: Master
       ```

     - On the replica server, apply the following modify:

       ```
       dn: cn=configuration
       changetype: modify
       replace: ibm-slapdserverid
       ibm-slapdserverid: Replica
       ```

  2. **Consumer server credentials entry:** Add this entry to the replica server using the **ldapadd** utility. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for additional information about the **ldapadd** utility. For example:

     ```
     dn: cn=Master server,cn=configuration
     changetype: add
     objectclass: ibm-slapdReplication
     cn: master server
     ibm-slapdMasterDN: cn=bindtoconsumer
     ibm-slapdMasterPW: iamsupplier
     ibm-slapdMasterReferral: ldap://server1.us.ibm.com:389
     ```

- The next step is to build the LDIF file for the topology. This LDIF file is called **masterreplica.ldif**. Copy each of these entries to **masterreplica.ldif** with the necessary changes in the subtree, server IDs, host names, and ports.
  1. **Replication context:**
     - If the subtree entry exists, use the **ldapmodify** utility to modify the existing entry. For example:

       ```
       dn: o=ibm,c=us
       changetype: modify
       add: objectclass
       objectclass: ibm-replicationContext
       ```

     - If the subtree entry does not exist, add the entry with the **ibm-replicationContext** auxiliary objectclass by using the **ldapadd** utility. For example:

       ```
       dn: o=ibm,c=us
       changetype: add
       objectclass: top
       objectclass: organization
       objectclass: ibm-replicationContext
       o: ibm
       ```

  2. Add the replica group entry using the **ldapadd** utility. For example:

     ```
     dn: ibm-replicaGroup=default, o=ibm,c=us
     changetype: add
     objectclass: top
     objectclass: ibm-replicaGroup
     ibm-replicaGroup: default
     ```

  3. **Replica subentries:** Because this topology is using a master and a read-only replica server, a replica subentry is only needed for the master server. Read-only replicas do not need a replica subentry. For example, for a subentry for the master:

     ```
     dn: ibm-replicaServerId=Master,ibm-replicaGroup=default, o=ibm,c=us
     changetype: add
     objectclass: top
     objectclass: ibm-replicaSubentry
     ibm-replicaServerId: Master
     ibm-replicationServerIsMaster: true
     cn: Master
     description: Master server of the topology
     ```

     Take note of the master subentry carefully. The subentry for the master uses the server ID `Master` and has the server declared as a master server. This server receives updates from clients.

     Note: The number of subentries is not dependent on the number of physical servers in the topology. Rather, it is dependent on the number and role of the LDAP servers in the topology.

  4. **Supplier server credentials entry:** This step defines the credentials that the master uses to bind to the replica. Add an entry with the **ldapadd** utility. For example:

     ```
     dn: cn=ReplicaBindCredentials, o=ibm,c=us
     changetype: add
     objectclass: ibm-replicationCredentialsSimple
     cn: ReplicaBindCredentials
     replicaBindDN: cn=bindtoconsumer
     replicaCredentials: iamsupplier
     description: Bind Credentials on master to bind to replica
     ```

Note the DN and password are the same as the pair that was added in the consumer server credentials entry section above. Add the entry with the **ldapadd** utility. As a result, updates to this object results in the server attempting to replicate.

5. **Replication agreements:** There is one supplier-consumer relationship in this advanced replication topology. The master supplies updates to the `o=ibm,c=us` subtree to the replica that consumes the changes. Therefore, there is only one agreement: From the master to the replica. Note that the number of agreements is dependent upon the number of supplier-consumer relationships in the topology. For example:

```
dn: cn=Replica, ibm-replicaServerId=Master,ibm-replicaGroup=default,o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Replica
ibm-replicaConsumerId: Replica
ibm-replicaUrl: ldap://server2.us.ibm.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=ibm,c=us
description: Replication agreement from master to replica
```

The entry above is under the subentry for the master. It supplies to a consumer with an ID as `Replica`. The replica URL is `ldap://server2.us.ibm.com:389` meaning that the replica is listening on `server2.us.ibm.com` on port `389`. This agreement uses the credentials that were created in the last step for the master to bind to the replica. That means the master binds to the replica with a bind DN `cn=bindtoconsumer` and the password `iamsupplier`. Note that there is no agreement under the subentry for the replica. This is natural as the replica is a read-only copy and cannot receive any client updates, therefore, there is no point in having an agreement, because there are no updates to propagate.

- Now that the replication entries have been added, the **masterreplica.ldif** file is as follows:

```
dn: o=ibm,c=us
changetype: add
objectclass: top
objectclass: organization
objectclass: ibm-replicationContext
o: ibm

dn: ibm-replicaGroup=default, o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicaGroup
ibm-replicaGroup: default

dn: ibm-replicaServerId=Master,ibm-replicaGroup=default, o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: Master
ibm-replicationServerIsMaster: true
cn: Master
description: Master server of the topology.

dn: cn=ReplicaBindCredentials, o=ibm,c=us
changetype: add
objectclass: ibm-replicationCredentialsSimple
cn: ReplicaBindCredentials
replicaBindDN: cn=bindtoconsumer
replicaCredentials: iamsupplier
description: Bind Credentials on master to bind to replica.

dn: cn=Replica, ibm-replicaServerId=Master,ibm-replicaGroup=default,o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Replica
ibm-replicaConsumerId: Replica
```

```
ibm-replicaUrl: ldap://server2.us.ibm.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=ibm,c=us
description: Replication agreement from master to replica.
```

The LDIF is different if the replication context exists. Rather than:

```
dn: o=ibm,c=us
changetype: add
objectclass: top
objectclass: organization
objectclass: ibm-replicationContext
o: ibm
```

it is:

```
dn: o=ibm,c=us
changetype: modify
add: objectclass
objectclass: ibm-replicationContext
```

For the sake of simplicity, it is assumed that the subtree entry does not exist at all.

- Next, add the **masterreplica.ldif** file on the master server. Use the **ldapadd** utility on the master where the **masterreplica.ldif** file was created. For example:

  ```
  ldapadd -h server1.us.ibm.com -p 389 -D adminDN -w adminPW -f masterreplica.ldif -k -L
  ```

  The **–L** option on the **ldapadd** utility sends the **Do Not Replicate** control to the server that indicates not to replicate the topology now. The **-k** option sends the **Server Administration** control to the server so that the addition of entries continues even when the subtree becomes read-only because of a server ID mismatch.
- Next add the replication topology to the replica also. Use the **ldapexop** utility. For example:

  ```
  ldapexop -h server1.us.ibm.com -p 389 -D adminDN -w adminPW -op repltopology
   -rc o=ibm,c=us
  ```

  This is an example of the **Replication topology** extended operation. It propagates the advanced replication topology to all the consumers defined under the `o=ibm,c=us` replication context. See “ldapexop utility” on page 255 for additional information about the **ldapexop** utility.
- After the **ldapadd** and **ldapexop** commands are performed successfully, the master-replica topology is ready. The master accepts updates on the `o=ibm,c=us` subtree and propagates them to the replica. The replica does not accept updates. It returns a referral to the master in case a client tries to update it, however, it can handle searches.

Creating a peer-to-peer replication topology

The peer-to-peer replication topology does not differ much from the master-replica topology. It also has two servers, but, both the servers are now read/write servers. They both supply changes to each other.

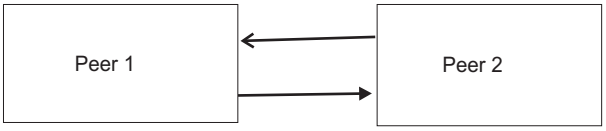


Figure 50. Peer-to-peer topology

- The first step when building a topology is to define:

1. **Replication context:** o=ibm,c=us
2. **Supplier(s):** LDAP server on server1.us.ibm.com:389 with server ID Peer1 supplies updates to the LDAP server with server ID Peer2 on server2.us.ibm.com:389. LDAP server on server2.us.ibm.com:389 with server ID Peer2 supplies updates to the LDAP server with server ID Peer1 on server1.us.ibm.com:389.
3. **Consumer(s):** LDAP server with server ID Peer2 on server2.us.ibm.com:389 consumes updates from LDAP server with server ID Peer1 on server1.us.ibm.com:389. LDAP server with Server ID Peer1 on server1.us.ibm.com:389 consumes updates from LDAP server with server ID Peer2 on server2.us.ibm.com:389.
4. **read/write server(s):** LDAP servers Peer1 and Peer2 on server1.us.ibm.com and server2.us.ibm.com are read/write servers.
5. **Read-only server(s):** There are no read-only servers in this topology.

Configuration changes: Some CDBM changes must be done to both the Peer1 and Peer2 servers for replication to work correctly. Configured servers are current. If you are using the same servers that you used for the Master-Replica setup, undo the changes that you made in "Creating a master-replica topology" on page 515.

Note: These are done here for this example only. The server ID is only allowed to be modified when there are no replica subentries defined in the server. See Table 10 on page 139 for more information about the **ibm-slapdServerID** attribute value.

1. **Server IDs:**
   - On the Peer1 server (server1.us.ibm.com), apply the following modify using the **ldapmodify** utility. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for additional information about the **ldapmodify** utility.

     ```
     dn: cn=configuration
     changetype: modify
     replace: ibm-slapdserverid
     ibm-slapdserverid: Peer1
     ```

   - On the Peer2 server (server2.us.ibm.com), apply the following modify using the **ldapmodify** utility.

     ```
     dn: cn=configuration
     changetype: modify
     replace: ibm-slapdserverid
     ibm-slapdserverid: Peer2
     ```

2. **Consumer server credentials entry:** Add this first entry to the Peer1 server and the second entry to the Peer2 server using the **ldapadd** utility. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for additional information about the **ldapadd** utility.

   Note: An alternative is to add entries with an **ibm-slapdSupplier** objectclass. Also, each peer uses the other peer as the referral. This is useful if a peer must become a read only replica or a forwarding server.
   For example, for a consumer side credentials entry for the Peer1:

   ```
   dn: cn=Master server,cn=configuration
   changetype: add
   objectclass: ibm-slapdReplication
   cn: master server
   ibm-slapdMasterDN: cn=bindtoconsumer
   ibm-slapdMasterPW: iamsupplier
   ibm-slapdMasterReferral: ldap://server2.us.ibm.com:389
   ```

For example, for a credential subentry for the Peer2:

```
dn: cn=Master server,cn=configuration
changetype: add
objectclass: ibm-slapdReplication
cn: master server
ibm-slapdMasterDN: cn=bindtoconsumer
ibm-slapdMasterPW: iamsupplier
ibm-slapdMasterReferral: ldap://server1.us.ibm.com:389
```

- The next step is to build the LDIF file for the replication topology. This LDIF file is called **peer2peer.ldif**. Copy each of these entries to **peer2peer.ldif** with the necessary changes in the subtree, server IDs, host names, and ports.
  1. **Replication context:**
     - If the subtree entry exists, use the **ldapmodify** utility to modify the existing entry. For example:

       ```
       dn: o=ibm,c=us
       changetype: modify
       add: objectclass
       objectclass: ibm-replicationContext
       ```

     - If the subtree entry does not exist, add the entry with the **ibm-replicationContext** auxiliary objectclass by using the **ldapadd** utility. For example:

       ```
       dn: o=ibm,c=us
       changetype: add
       objectclass: top
       objectclass: organization
       objectclass: ibm-replicationContext
       o: ibm
       ```

  2. Add the replica group entry using the **ldapadd** utility. For example:

     ```
     dn: ibm-replicaGroup=default, o=ibm,c=us
     changetype: add
     objectclass: top
     objectclass: ibm-replicaGroup
     ibm-replicaGroup: default
     ```

  3. **Replica subentries:** Because there are two LDAP servers in the topology, you must add two replica subentries; one for Peer1 and another one for the Peer2 server by using the **ldapadd** utility. For example, for a subentry for Peer1:

     ```
     dn: ibm-replicaServerId=Peer1,ibm-replicaGroup=default, o=ibm,c=us
     changetype: add
     objectclass: top
     objectclass: ibm-replicaSubentry
     ibm-replicaServerId: Peer1
     ibm-replicationServerIsMaster: true
     cn: Peer1
     description: Subentry for Peer1
     ```

     For example, for a subentry for Peer2:

     ```
     dn: ibm-replicaServerId=Peer2,ibm-replicaGroup=default, o=ibm,c=us
     changetype: add
     objectclass: top
     objectclass: ibm-replicaSubentry
     ibm-replicaServerId: Peer2
     ibm-replicationServerIsMaster: true
     cn: Peer2
     description: Subentry for Peer2
     ```

     The subentry for Peer1 identifies the server ID as Peer1 and has the server declared as a master server. The server can receive updates from clients. The subentry for the Peer2 identifies the server ID as Peer2, and it also has the server declared as a master server. This server can also receive updates from LDAP clients that bind to it.

4. **Supplier server credentials entry:** This step defines the credentials that Peer1 and Peer2 use to bind to each other. Add the entry with the **ldapadd** utility. For example:

```
dn: cn=ReplicaBindCredentials, o=ibm,c=us
changetype: add
objectclass: ibm-replicationCredentialsSimple
cn: ReplicaBindCredentials
replicaBindDN: cn=bindtoconsumer
replicaCredentials: iamsupplier
description: Bind Credentials on Peer1 and Peer2 to bind to each other.
```

5. **Replication agreements:** There are two supplier-consumer relationships in this topology. Peer1 supplies updates made to the o=ibm,c=us subtree to Peer2, that consumes the changes. Peer2 also accepts updates from clients on the o=ibm,c=us subtree and sends them to Peer1, that consumes the changes. Therefore, there are two agreements:

   a. from Peer1 to Peer2

   b. from Peer2 to Peer1

   Note: The number of agreements is dependent upon the number of supplier-consumer relationships in the topology.
   For example, a replication agreement from Peer1 to Peer2:

```
dn: cn=Peer2, ibm-replicaServerId=Peer1,ibm-replicaGroup=default,o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Peer2
ibm-replicaConsumerId: Peer2
ibm-replicaUrl: ldap://server2.us.ibm.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=ibm,c=us
description: Replication agreement from Peer1 to Peer2
```

   For example, a replication agreement from Peer2 to Peer1:

```
dn: cn=Peer1, ibm-replicaServerId=Peer2,ibm-replicaGroup=default,o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Peer1
ibm-replicaConsumerId: Peer1
ibm-replicaUrl: ldap://server1.us.ibm.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=ibm,c=us
description: Replication agreement from Peer2 to Peer1
```

   The two agreements for this topology are shown above. The first one is the agreement from Peer1 to Peer2. It is under the Peer1 subentry. The second one is from Peer2 to Peer1 and it is under the Peer2 subentry. Note the use of the same credentials entry for both agreements. This is acceptable. The credentials entry is added to both Peer1 and Peer2.

- Now that the replication entries have been added, the **peer2peer.ldif** file is as follows:

```
dn: o=ibm,c=us
changetype: add
objectclass: top
objectclass: organization
objectclass: ibm-replicationContext
o: ibm

dn: ibm-replicaGroup=default, o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicaGroup
ibm-replicaGroup: default
```

```
dn: ibm-replicaServerId=Peer1,ibm-replicaGroup=default, o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: Peer1
ibm-replicationServerIsMaster: true
cn: Peer1
description: Subentry for Peer1.

dn: ibm-replicaServerId=Peer2,ibm-replicaGroup=default, o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: Peer2
ibm-replicationServerIsMaster: true
cn: Peer2
description: Subentry for Peer2.

dn: cn=ReplicaBindCredentials, o=ibm,c=us
changetype: add
objectclass: ibm-replicationCredentialsSimple
cn: ReplicaBindCredentials
replicaBindDN: cn=bindtoconsumer
replicaCredentials: iamsupplier
description: Bind Credentials on Peer1 and Peer2 to bind to each other.

dn: cn=Peer2, ibm-replicaServerId=Peer1,ibm-replicaGroup=default,o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Peer2
ibm-replicaConsumerId: Peer2
ibm-replicaUrl: ldap://server2.us.ibm.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=ibm,c=us
description: Replication agreement from Peer1 to Peer2.

dn: cn=Peer1, ibm-replicaServerId=Peer2,ibm-replicaGroup=default,o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Peer1
ibm-replicaConsumerId: Peer1
ibm-replicaUrl: ldap://server1.us.ibm.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=ibm,c=us
description: Replication agreement from Peer2 to Peer1
```

- Next, add the **peer2peer.ldif** file on the Peer1 server. Use the **ldapadd** utility on the Peer1 where the **peer2peer.ldif** file was created. For example:

  ```
  ldapadd -h server1.us.ibm.com -p 389 -D  adminDN -w adminPW -f peer2peer.ldif -k -L
  ```

  The **–L** option on the **ldapadd** utility sends the **Do Not Replicate** control to the server that indicates not to replicate the topology now. The **-k** option sends the **Server Administration** control to the server so that the addition of entries continues even when the subtree becomes read-only because of a server ID mismatch.

- Next, add the replication topology to the Peer2 also. Use the **ldapexop** utility. For example:

  ```
  ldapexop -h server1.us.ibm.com -p 389 -D  adminDN -w adminPW -op repltopology
   -rc o=ibm,c=us
  ```

  This is an example of the **Replication topology** extended operation. It propagates the advanced replication topology to all the consumers defined under the o=ibm,c=us replication context. See “ldapexop utility” on page 255 for additional information about the **ldapexop** utility.

- After the **ldapadd** and **ldapexop** commands are performed successfully, the peer-to-peer topology is ready. Both peers accept updates on and send them to the other peer.

Creating a master-forwarder-replica (cascading) topology

Another advanced replication topology is the master-forwarder-replica or cascading replication topology.

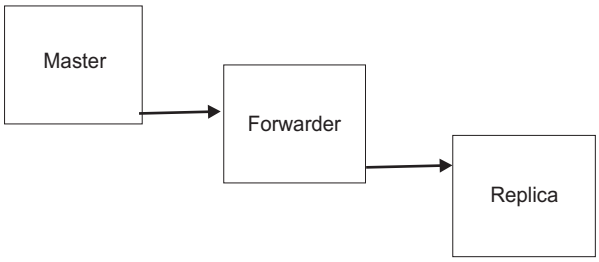


Figure 51. Master-forwarder-replica topology

The forwarder server is a specialized replica server. As previously stated, replica servers are read-only as is a forwarder server. Replica servers do not transmit changes that are consumed by them. However, forwarder servers replicate changes that they have consumed. To supply changes further down the topology, forwarders have agreements under their subentries.

Note: Gateway replication topologies are similar, however, forwarders are specialized replicas while gateways are specialized masters.

This topology must have one more server included: `server3.us.ibm.com`

- The first step when building a topology is to define:
  1. **Replication context:** `o=ibm,c=us`
  2. **Supplier(s):** LDAP server on `server1.us.ibm.com:389` with server ID `Master` supplies updates to the LDAP server with server ID `Forwarder` on `server2.us.ibm.com:389`. LDAP server on `server2.us.ibm.com:389` with server ID `Forwarder` supplies updates to the LDAP server with server ID `Replica` on `server3.us.ibm.com:389`.
  3. **Consumer(s):** LDAP server with server ID `Forwarder` on `server2.us.ibm.com:389` consumes updates from LDAP server with server ID `Master` working on `server1.us.ibm.com:389`. LDAP server with server ID `Replica` on `server3.us.ibm.com:389` consumes updates from the LDAP server with server ID `Forwarder` working on `server2.us.ibm.com:389`.
  4. **read/write server(s):** LDAP server on `server1.us.ibm.com:389`, with ID `Master` is the only read/write server.
  5. **Read-only server(s):** LDAP server on `server2.us.ibm.com:389` and `server3.us.ibm.com:389`, with IDs, `Forwarder` and `Replica` are read-only.

  Configuration changes: Some configuration changes must be made to the `Master`, `Forwarder`, and `Replica` servers for replication to work correctly. Configured servers are current. If reusing the same servers that were used for the Master-Replica setup, undo the changes that were made in "Creating a master-replica topology" on page 515. If reusing the same servers that were used for the Peer-Peer setup, undo the changes that were made in "Creating a peer-to-peer replication topology" on page 518.

  Note: These are done here for this example only. The server ID is only allowed to be modified when there are no replica subentries defined in the server. See Table 10 on page 139 for more information about the **ibm-slapdServerID** attribute value.

1. **Server IDs:**
   - On the Master server (server1.us.ibm.com), apply the following modify using the **ldapmodify** utility. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for additional information about the **ldapmodify** utility.

     ```
     dn: cn=configuration
     changetype: modify
     replace: ibm-slapdserverid
     ibm-slapdserverid: Master
     ```

   - On the Forwarder server (server2.us.ibm.com), apply the following modify using the **ldapmodify** utility.

     ```
     dn: cn=configuration
     changetype: modify
     replace: ibm-slapdserverid
     ibm-slapdserverid: Forwarder
     ```

   - On the Replica server (server3.us.ibm.com), apply the following modify using the **ldapmodify** utility.

     ```
     dn: cn=configuration
     changetype: modify
     replace: ibm-slapdserverid
     ibm-slapdserverid: Replica
     ```

2. **Consumer server credentials entry:** Add this first entry to the Forwarder server and the second entry to the Replica server using the **ldapadd** utility. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for additional information about the **ldapadd** utility.

   Note: An alternative is to add entries with an **ibm-slapdSupplier** objectclass.
   For example, for a consumer server credentials entry for the Forwarder:

   ```
   dn: cn=Master server,cn=configuration
   changetype: add
   cn: master server
   ibm-slapdMasterDN: cn=bindtoconsumer
   ibm-slapdMasterPW: iamsupplier
   ibm-slapdMasterReferral: ldap://server1.us.ibm.com:389
   objectclass: ibm-slapdReplication
   ```

   For example, for a consumer server credentials entry for the Replica:

   ```
   dn: cn=Master server,cn=configuration
   changetype: add
   cn: master server
   ibm-slapdMasterDN: cn=bindtoconsumer
   ibm-slapdMasterPW: iamsupplier
   ibm-slapdMasterReferral: ldap://server1.us.ibm.com:389
   objectclass: ibm-slapdReplication
   ```

- The next step is to build the LDIF file for the replication topology. This LDIF file is called **mfr.ldif**. Copy each of these entries to **mfr.ldif** with the necessary changes in the subtree, server IDs, host names, and ports.

  1. **Replication context:**
     - If the subtree entry exists, use the **ldapmodify** utility to modify the existing entry. For example:

       ```
       dn: o=ibm,c=us
       changetype: modify
       add: objectclass
       objectclass: ibm-replicationContext
       ```

– If the subtree entry does not exist, add the entry with the **ibm-replicationContext** auxiliary objectclass by using the **ldapadd** utility. For example:

```
dn: o=ibm,c=us
changetype: add
objectclass: top
objectclass: organization
objectclass: ibm-replicationContext
o: ibm
```

2. Add the replica group entry using the **ldapadd** utility. For example:

```
dn: ibm-replicaGroup=default, o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicaGroup
ibm-replicaGroup: default
```

3. **Replica subentries:** Because this topology is using a master, a forwarding, and a read-only replica server, replica subentries are only needed for the `Master` and for the `Forwarder` servers. The read-only `Replica` server does not need a replica subentry. The following entries can be added using the **ldapadd** utility.

   For example, for a subentry for `Master`:

```
dn: ibm-replicaServerId=Master,ibm-replicaGroup=default, o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: Master
ibm-replicationServerIsMaster: true
cn: Master
description: Subentry for Master
```

   For example, for a subentry for `Forwarder`:

```
dn: ibm-replicaServerId=Forwarder,ibm-replicaGroup=default, o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: Forwarder
ibm-replicationServerIsMaster: false
cn: Forwarder
description: Subentry for the Forwarder
```

   The subentry for `Master` identifies the server ID as `Master` and has the server declared as a master server. The server can receive updates from clients. The subentry for the `Forwarder` identifies the server ID as `Forwarder`, and is declared as a non-master server. It cannot get updates from clients.

4. **Supplier server credentials entry:** This step defines the credentials that the `Master` uses to bind to the `Forwarder`. Add the entry with the **ldapadd** utility. For example:

```
dn: cn=ReplicaBindCredentials, o=ibm,c=us
changetype: add
objectclass: ibm-replicationCredentialsSimple
cn: ReplicaBindCredentials
replicaBindDN: cn=bindtoconsumer
replicaCredentials: iamsupplier
description: Bind Credentials on master to bind to forwarder
```

5. **Replication agreements:** There are two supplier-consumer relationships in this topology. `Master` supplies updates made to the `o=ibm,c=us` subtree to `Forwarder`, that consumes the changes and then supplies these changes to the `Replica`. Therefore, there are two agreements:

   a. from `Master` to `Forwarder`

b. from Forwarder to Replica

Note: The number of replication agreements is dependent upon the number of supplier-consumer relationships in the topology.
For example, a replication agreement from Master to Forwarder:

```
dn: cn=Forwarder, ibm-replicaServerId=Master,ibm-replicaGroup=default,o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Forwarder
ibm-replicaConsumerId: Forwarder
ibm-replicaUrl: ldap://server2.us.ibm.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=ibm,c=us
description: Replication agreement from Master to Forwarder
```

For example, a replication agreement from Forwarder to Replica:

```
dn: cn=Replica, ibm-replicaServerId=Forwarder,ibm-replicaGroup=default,o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Replica
ibm-replicaConsumerId: Replica
ibm-replicaUrl: ldap://server3.us.ibm.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=ibm,c=us
description: Replication agreement from Forwarder to Replica
```

The two agreements for this topology are shown above. The first one is the agreement from Master to Forwarder. It is under the Master subentry. The second one is from Forwarder to Replica and it is under the Forwarder subentry. Note the use of the same credentials entry for both agreements. This is acceptable. The consumer server credentials entry is added to both Master and Forwarder servers.

- Now that the replication entries have been added, the **mfr.ldif** file is as follows:

```
dn: o=ibm,c=us
changetype: add
objectclass: top
objectclass: organization
objectclass: ibm-replicationContext
o: ibm

dn: ibm-replicaGroup=default, o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicaGroup
ibm-replicaGroup: default

dn: ibm-replicaServerId=Master,ibm-replicaGroup=default, o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: Master
ibm-replicationServerIsMaster: true
cn: Master
description: Subentry for Master.

dn: ibm-replicaServerId=Forwarder,ibm-replicaGroup=default, o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: Forwarder
ibm-replicationServerIsMaster: false
cn: Forwarder
description: Subentry for the Forwarder.

dn: cn=ReplicaBindCredentials, o=ibm,c=us
changetype: add
objectclass: ibm-replicationCredentialsSimple
cn: ReplicaBindCredentials
replicaBindDN: cn=bindtoconsumer
replicaCredentials: iamsupplier
description: Bind Credentials on master to bind to forwarder.
```

```
dn: cn=Forwarder, ibm-replicaServerId=Master,ibm-replicaGroup=default,o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Forwarder
ibm-replicaConsumerId: Forwarder
ibm-replicaUrl: ldap://server2.us.ibm.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=ibm,c=us
description: Replication agreement from Master to Forwarder

dn: cn=Replica, ibm-replicaServerId=Forwarder,ibm-replicaGroup=default,o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Replica
ibm-replicaConsumerId: Replica
ibm-replicaUrl: ldap://server3.us.ibm.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=ibm,c=us
description: Replication agreement from Forwarder to Replica
```

- Next, add the **mfr.ldif** file on the `Master` server. Use the **ldapadd** utility on the `Master` where the **mfr.ldif** file was created. For example:

  ```
  ldapadd -h server1.us.ibm.com -p 389 -D adminDN -w adminPW -f mfr.ldif -k -L
  ```

- Next, add the **mfr.ldif** file on the `Forwarder` server. Use the **ldapadd** utility on the `Master` server and target the `Forwarder` server. For example:

  ```
  ldapadd -h server2.us.ibm.com -p 389 -D adminDN -w adminPW -f mfr.ldif -k -L
  ```

  The **–L** option on the **ldapadd** utility sends the **Do Not Replicate** control to the server that indicates not to replicate the topology now. The **-k** option sends the **Server Administration** control to the server so that the addition of entries continues even when the subtree becomes read-only because of a server ID mismatch.

- Next, add the replication topology to the `Replica` also. Use the **ldapexop** utility. For example:

  ```
  ldapexop -h server2.us.ibm.com -p 389 -D adminDN -w adminPW -op repltopology -rc
   o=ibm,c=us
  ```

  This is an example of the **Replication topology** extended operation. It propagates the advanced replication topology to the `Replica` defined under the `o=ibm,c=us` replication context. See “ldapexop utility” on page 255 for additional information about the **ldapexop** utility.

- After the **ldapadd** and **ldapexop** commands are performed successfully, the master-forwarder-replica topology is ready. The `Master` accepts updates, that go to the `Forwarder` and then to the `Replica`. If you intend to add another replica to the topology, under the forwarder, you must add another subentry for the replica, and add an agreement from the forwarder to that replica.

Creating a gateway topology

A gateway topology requires at least two gateway servers. Gateway topologies are created in a similar manner to a master-forwarder-replica (cascading) topology. A gateway topology includes gateway master servers that forward all replication traffic from the local replication site where it exists to other gateway servers in the replicating network. A gateway server also receives replication traffic from other gateway servers within the replication network and forwards updates to all servers on its local replication site.

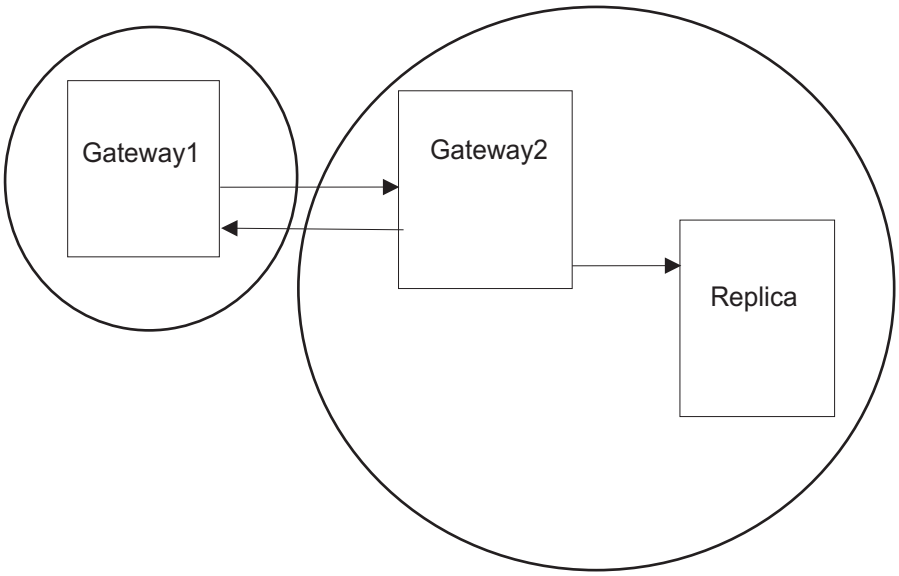


Figure 52. Gateway topology

In a gateway topology, gateway servers are distinguished from normal master servers by including the **ibm-replicaGateway** objectclass for the replica subentry in the replication context. As previously stated, a server is a master if the server ID in the replica subentry within the replication context equals the servers server ID and the subentrys **ibm-replicationServerIsMaster** attribute is set to `true`. See "Replica subentries" on page 497 for more information about replica subentries.

- The first step when building a topology is to define:
  1. **Replication context:** `o=ibm,c=us`
  2. **Supplier(s):** LDAP server on `server1.us.ibm.com:389` with server ID `Gateway1` supplies updates to the LDAP server with server ID `Gateway2` on `server2.us.ibm.com:389`. LDAP server on `server2.us.ibm.com:389` with server ID `Gateway2` supplies updates to the LDAP server with server ID `Gateway1` on `server1.us.ibm.com:389` and to the LDAP server with server ID `Replica` on `server3.us.ibm.com:389`.
  3. **Consumer(s):** LDAP server with server ID `Gateway2` on `server2.us.ibm.com:389` consumes updates from LDAP server with server ID `Gateway1` on `server1.us.ibm.com:389`. LDAP server with Server ID `Gateway1` on `server1.us.ibm.com:389` consumes updates from LDAP server with server ID `Gateway2` on `server2.us.ibm.com:389`. LDAP server with server ID `Replica` on `server3.us.ibm.com:389` consumes updates from LDAP server with server ID `Gateway2` on `server2.us.ibm.com:389`.
  4. **read/write server(s):** LDAP servers `Gateway1` and `Gateway2` on `server1.us.ibm.com` and `server2.us.ibm.com` are read/write servers.
  5. **Read-only server(s):** LDAP server `Replica` on `server3.us.ibm.com` is a read-only server.

  Configuration changes: Some configuration changes must be made to the `Gateway1`, `Gateway2`, and `Replica` servers for replication to work correctly. Configured servers should be current. If reusing the same servers that were used for the Master-Replica setup, undo the changes that were made in "Creating a master-replica topology" on page 515. If reusing the same servers that were used for the Peer-Peer setup, undo the changes that were made in "Creating a peer-to-peer replication topology" on page 518. If reusing the same servers that

were used for the Master-forwarder-replica setup, undo the changes that were made in "Creating a master-forwarder-replica (cascading) topology" on page 523.

Note: These are done here for this example only. The server ID is only allowed to be modified when there are no replica subentries defined in the server. See Table 10 on page 139 for more information about the **ibm-slapdServerID** attribute value.

1. **Server IDs:**
   - On the Gateway1 server (server1.us.ibm.com), apply the following modify using the **ldapmodify** utility. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for additional information about the **ldapmodify** utility.

     ```
     dn: cn=configuration
     changetype: modify
     replace: ibm-slapdserverid
     ibm-slapdserverid: Gateway1
     ```

   - On the Gateway2 server (server2.us.ibm.com), apply the following modify using the **ldapmodify** utility.

     ```
     dn: cn=configuration
     changetype: modify
     replace: ibm-slapdserverid
     bm-slapdserverid: Gateway2
     ```

   - On the Replica server (server3.us.ibm.com), apply the following modify using the **ldapmodify** utility.

     ```
     dn: cn=configuration
     changetype: modify
     replace: ibm-slapdserverid
     ibm-slapdserverid: Replica
     ```

2. **Consumer server credentials entry:** Add the first entry to the Gateway1 server, the second entry to the Gateway2 server, and the third entry to the Replica server using the **ldapadd** utility. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for additional information about the **ldapadd** utility.

   Note: An alternative is to add entries with an **ibm-slapdSupplier** objectclass.

   For example, for a consumer server credentials entry for Gateway1:

   ```
   dn: cn=Master server,cn=configuration
   changetype: add
   cn: master server
   ibm-slapdMasterDN: cn=bindtoconsumer
   ibm-slapdMasterPW: iamsupplier
   ibm-slapdMasterReferral: ldap://server2.us.ibm.com:389
   objectclass: ibm-slapdReplication
   ```

   For example, for a consumer server credentials entry for Gateway2:

   ```
   dn: cn=Master server,cn=configuration
   changetype: add
   cn: master server
   ibm-slapdMasterDN: cn=bindtoconsumer
   ibm-slapdMasterPW: iamsupplier
   ibm-slapdMasterReferral: ldap://server1.us.ibm.com:389
   objectclass: ibm-slapdReplication
   ```

   For example, for a consumer server credentials entry for Replica:

   ```
   dn: cn=Master server,cn=configuration
   changetype: add
   cn: master server
   ```

```
ibm-slapdMasterDN: cn=bindtoconsumer
ibm-slapdMasterPW: iamsupplier
ibm-slapdMasterReferral: ldap://server2.us.ibm.com:389
objectclass: ibm-slapdReplication
```

- The next step is to build the LDIF file for the replication topology. This LDIF file is called **gateway.ldif**. Copy each of these entries to **gateway.ldif** with the necessary changes in the subtree, server IDs, host names, and ports.
  1. **Replication context:**
     - If the subtree entry exists, use the **ldapmodify** utility to modify the existing entry. For example:

       ```
       dn: o=ibm,c=us
       changetype: modify
       add: objectclass
       objectclass: ibm-replicationContext
       ```

     - If the subtree entry does not exist, add the entry with the **ibm-replicationContext** auxiliary objectclass by using the **ldapadd** utility. For example:

       ```
       dn: o=ibm,c=us
       changetype: add
       objectclass: top
       objectclass: organization
       objectclass: ibm-replicationContext
       o: ibm
       ```

  2. Add the replica group entry using the **ldapadd** utility. For example:

     ```
     dn: ibm-replicaGroup=default, o=ibm,c=us
     changetype: add
     objectclass: top
     objectclass: ibm-replicaGroup
     ibm-replicaGroup: default
     ```

  3. **Replica subentries:** Because this topology is using two gateways and a read-only replica server, replica subentries are only needed for the `Gateway1` and for the `Gateway2` servers. The read-only `Replica` server does not need a replica subentry. The following entries can be added using the **ldapadd** utility.

     For example, for a subentry for `Gateway1` (note that an objectclass of **ibm-replicaGateway** has been added to this entry to indicate that it is a gateway server):

     ```
     dn: ibm-replicaServerId=Gateway1,ibm-replicaGroup=default, o=ibm,c=us
     changetype: add
     objectclass: top
     objectclass: ibm-replicaSubentry
     objectclass: ibm-replicaGateway
     ibm-replicaServerId: Gateway1
     ibm-replicationServerIsMaster: true
     cn: Gateway1
     description: Subentry for Gateway1.
     ```

     For example, for a subentry for `Gateway2` (note that an objectclass of **ibm-replicaGateway** has been added to this entry to indicate that it is a gateway server):

     ```
     dn: ibm-replicaServerId=Gateway2,ibm-replicaGroup=default, o=ibm,c=us
     changetype: add
     objectclass: top
     objectclass: ibm-replicaSubentry
     objectclass: ibm-replicaGateway
     ibm-replicaServerId: Gateway2
     ibm-replicationServerIsMaster: true
     cn: Gateway2
     description: Subentry for Gateway2.
     ```

The subentry for Gateway1 identifies the server ID as Gateway1 and has the server declared as a gateway server. The Gateway1 server can receive updates from clients. The subentry for Gateway2 identifies the server ID as Gateway2 and has the server declared as a gateway server. The Gateway2 server can receive updates from clients.

4. **Supplier server credentials entry:** This step defines the credentials that Gateway1 uses to bind to Gateway2, Gateway2 uses to bind to Gateway1, and Gateway2 uses to bind to Replica. Add the entry with the **ldapadd** utility. For example:

```
dn: cn=ReplicaBindCredentials, o=ibm,c=us
changetype: add
objectclass: ibm-replicationCredentialsSimple
cn: ReplicaBindCredentials
replicaBindDN: cn=bindtoconsumer
replicaCredentials: iamsupplier
description: Bind Credentials used on the gateway servers.
```

5. **Replication agreements:** There are three supplier-consumer relationships in this topology. Gateway1 supplies updates made to the o=ibm,c=us subtree to Gateway2. Gateway2 supplies updates made to the o=ibm,c=us subtree to Gateway1 and to Replica. Therefore, there are three agreements:

   a. from Gateway1 to Gateway2
   b. from Gateway2 to Gateway1
   c. from Gateway2 to Replica

   Note: The number of replication agreements is dependent upon the number of supplier-consumer relationships in the topology.

   For example, a replication agreement from Gateway1 to Gateway2:

```
dn: cn=Gateway2,ibm-replicaServerId=Gateway1,ibm-replicaGroup=default,o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Gateway2
ibm-replicaConsumerId: Gateway2
ibm-replicaUrl: ldap://server2.us.ibm.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=ibm,c=us
description: Replication agreement from Gateway1 to Gateway2.
```

   For example, a replication agreement from Gateway2 to Gateway1:

```
dn: cn=Gateway1,ibm-replicaServerId=Gateway2,ibm-replicaGroup=default,o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Gateway1
ibm-replicaConsumerId: Gateway1
ibm-replicaUrl: ldap://server1.us.ibm.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=ibm,c=us
description: Replication agreement from Gateway2 to Gateway1.
```

   For example, a replication agreement from Gateway2 to Replica:

```
dn: cn=Replica, ibm-replicaServerId=Gateway2,ibm-replicaGroup=default,o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Replica
ibm-replicaConsumerId: Replica
ibm-replicaUrl: ldap://server3.us.ibm.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=ibm,c=us
description: Replication agreement from Gateway2 to Replica.
```

   The three agreements for this topology are shown above. The first one is the agreement from Gateway1 to Gateway2. It is under the Gateway1 subentry. The second one is from Gateway2 to Gateway1 and it is under the Gateway2 subentry. The third one is from Gateway2 to Replica and it is also under the

Gateway2 subentry. Note the use of the same credentials entry for all three agreements. This is acceptable. The consumer server credentials entry is added to Gateway1, Gateway2, and Replica servers.

- Now that the replication entries have been added, the **gateway.ldif** file is as follows:

```
dn: o=ibm,c=us
changetype: add
objectclass: top
objectclass: organization
objectclass: ibm-replicationContext
o: ibm

dn: ibm-replicaGroup=default, o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicaGroup
ibm-replicaGroup: default

dn: ibm-replicaServerId=Gateway1,ibm-replicaGroup=default, o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicaSubentry
objectclass: ibm-replicaGateway
ibm-replicaServerId: Gateway1
ibm-replicationServerIsMaster: true
cn: Gateway1
description: Subentry for Gateway1.

dn: ibm-replicaServerId=Gateway2,ibm-replicaGroup=default, o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicaSubentry
objectclass: ibm-replicaGateway
ibm-replicaServerId: Gateway2
ibm-replicationServerIsMaster: true
cn: Gateway2
description: Subentry for Gateway2.

dn: cn=replicaBindCredentials, o=ibm,c=us
changetype: add
objectclass: ibm-replicationCredentialsSimple
cn: replicaBindCredentials
replicaBindDN: cn=bindtoconsumer
replicaCredentials: iamsupplier
description: Bind Credentials used on the gateway servers.

dn: cn=Gateway2,ibm-replicaServerId=Gateway1,ibm-replicaGroup=default,o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Gateway2
ibm-replicaConsumerId: Gateway2
ibm-replicaUrl: ldap://server2.us.ibm.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=ibm,c=us
description: Replication agreement from Gateway1 to Gateway2.

dn: cn=Gateway1,ibm-replicaServerId=Gateway2,ibm-replicaGroup=default,o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Gateway1
ibm-replicaConsumerId: Gateway1
ibm-replicaUrl: ldap://server1.us.ibm.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=ibm,c=us
description: Replication agreement from Gateway2 to Gateway1.

dn: cn=Replica, ibm-replicaServerId=Gateway2,ibm-replicaGroup=default,o=ibm,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Replica
```

```
ibm-replicaConsumerId: Replica
ibm-replicaUrl: ldap://server3.us.ibm.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=ibm,c=us
description: Replication agreement from Gateway2 to Replica.
```

- Next, add the **gateway.ldif** file on the `Gateway2` server. Use the **ldapadd** utility on `Gateway2` where the **gateway.ldif** file was created. For example:

  ```
  ldapadd -h server2.us.ibm.com -p 389 -D adminDN -w adminPW -f gateway.ldif -k -L
  ```

  The **-L** option on the **ldapadd** utility sends the **Do Not Replicate** control to the server that indicates not to replicate the topology now. The **-k** option sends the **Server Administration** control to the server so that the addition of entries continues even when the subtree becomes read-only because of a server ID mismatch.
- Next, add the replication topology to the `Gateway1` and `Replica` servers. Use the **ldapexop** utility. For example:

  ```
  ldapexop -h server2.us.ibm.com -p 389 -D adminDN -w adminPW -op repltopology
   -rc o=ibm,c=us
  ```

  This is an example of the **Replication topology** extended operation. It propagates the advanced replication topology to the `Gateway1` and `Replica` servers defined under the `o=ibm,c=us` replication context. See "ldapexop utility" on page 255 for more information about the **ldapexop** utility.
- After the **ldapadd** and **ldapexop** commands are performed successfully, the gateway topology is ready. The `Gateway1` server accepts updates that go to the `Gateway2` server. The `Gateway2` server also accepts updates that are then forwarded to the `Gateway1` and `Replica` servers.

Replication topology hints and tips

The following is helpful information about replication topologies.

1. When setting up the advanced replication topologies, all master servers are in maintenance mode until all the topology entries have been loaded on all servers participating in the topology. The **Do Not Replicate** and **Server Administration** controls are also used when adding entries for configuring advanced replication topologies. This precludes updates to master servers that would be lost if they are received before all server responsibilities (replica, master, forwarder, gateway) and relationships (replication agreement) being established.
2. All peers in a topology must supply to every other server in the topology unless they are separated by gateways. If they are separated by gateways, all the peers under a gateway must supply to all other servers including the gateway. This is because peers do not replicate changes supplied by other peers. That leads to peers receiving the updates they initiated.
3. All gateways in a topology must supply to each other. There must be at least two gateways in a topology for them to be useful.
4. Read-only servers do not accept updates that clients send. When an update is attempted against a read-only server, the referral list returned to the client is established from the following:
   a. The **ibm-replicationContext** objectclass allows for an optional attribute, **ibm-replicaReferralURL**. As stated previously, the **ibm-replicationContext** auxiliary objectclass must be added to the root of the subtree. This objectclass identifies the subtrees that are replication contexts. The **ibm-replicaReferralURL** attribute can hold a space delimited list of LDAP URLs. The URLs specified appears first in the list of referrals returned to the client. See "Replication contexts" on page 496 for more information about replication contexts.

b. The **cn=configuration** subtree in the CDBM backend allows a consumer server credentials entry with an objectclass of **ibm-slapdReplication** to be stored. If this object exists and contains a value for the **ibm-slapdMasterReferral** attribute, the value is appended to referral list set by the replication context. If the replication context does not define a referral list with the **ibm-replicaReferralURL** attribute, this is the only value sent to the client. See "Consumer server entries" on page 506 for more information about consumer server entries.

c. If the LDAP server configuration file has a **referral** configuration option specified and there are no consumer server credentials entries in the **cn=configuration** subtree with an **ibm-slapdMasterReferral** value, the **referral** option values are appended to the referral list set by the replication context. If the replication context does not have a referral list specified with the **ibm-replicaReferralURL** attribute and the consumer server credentials entry is not providing a referral list, the **referral** option is the only value sent to the client. See Chapter 8, "Customizing the LDAP server configuration," on page 83 for more information about the **referral** configuration option.

Replication of schema and password policy updates

Advanced replication allows the replication of schema, password policy entries, and other entries under the **cn=ibmpolicies** suffix. Schema, password policy, and other updates can be replicated by configuring a replication topology under the **cn=ibmpolicies** suffix in the CDBM backend. By default, schema and password policy updates are not replicated unless a replication topology is configured in the **cn=ibmpolicies** suffix.

Before configuring schema replication, verify that the schema between the servers is already synchronized by using the **ldapdiff** utility. See "ldapdiff utility" on page 247 for more information about the **ldapdiff** utility. If using the **ldapdiff** utility for schema comparison, the **-S** option must be specified. Make sure the **-L** option is specified so that schema differences are stored in an output LDIF file. The **ldapdiff** utility does not automatically fix schema differences on the consumer server. The schema on the consumer server must be manually modified with the output schema LDIF file generated by the **ldapdiff** utility.

Before configuring replication of password policy and other entries in the **cn=ibmpolicies** suffix, verify that the entries are already synchronized by using the **ldapdiff** utility. If LDAP password policy is active on both servers, make sure that each server is configured to use the same password policy rules.

Schema replication and replication of entries in the **cn=ibmpolicies** suffix is the same as configuring advanced replication in the LDBM or TDBM backends. See "Advanced replication configuration examples" on page 511 for information about configuring advanced replication, however, change the suffix used in those examples with **cn=ibmpolicies**. When the advanced replication entries are properly configured in the CDBM backend, the server performs schema replication and replication of entries in the **cn=ibmpolicies** suffix.

Protecting replication topology entries

The default propagating ACLs inherited from a suffix or root entry might be inappropriate for controlling access to the replication topology entries. To protect access to all replication topology entries in the server, make sure that the **ibm-slapdReplRestrictedAccess** attribute value is set to true in the

cn=replication,cn=configuration entry. When the **ibm-slapdReplRestrictedAccess** attribute is true, only an LDAP root, directory data, or replication administrator, and the master server DN for the replication context is allowed access to the replication topology entries. See Table 11 on page 141 for more information about the **ibm-slapdReplRestrictedAccess** attribute.

Unconfiguring advanced replication

If advanced replication is no longer needed, perform the following steps on the supplier server to remove the replication topology entries:

1. Ensure that the supplier server is in maintenance mode. See "Advanced replication maintenance mode" on page 536 for more information about advanced replication maintenance mode.
2. Bind to the supplier server as an LDAP administrator. See Chapter 9, "Administrative group and roles," on page 159 for more information about administrative role authority.
3. Delete the replication agreement entry.
4. Delete the replica subentry.
5. Delete the replica group.
6. Delete the **ibm-replicationContext** attribute value from the objectclass attribute type from the replication context.
7. Move the supplier server out of maintenance mode. See "Advanced replication maintenance mode" on page 536 for more information about advanced replication maintenance mode.

On the consumer server, perform the following steps to remove the replication topology entries:

1. Ensure that the consumer server is in maintenance mode. See "Advanced replication maintenance mode" on page 536 for more information about advanced replication maintenance mode.
2. Bind to the consumer server as an LDAP administrator. See Chapter 9, "Administrative group and roles," on page 159 for more information about administrative role authority.
3. Delete the replication agreement entry.
4. Delete the replica subentry.
5. Delete the replica group.
6. Delete the **ibm-replicaReferralURL** attribute from the replication context.
7. Delete the **ibm-replicationContext** attribute value from the objectclass attribute type from the replication context.
8. If the server is no longer a consumer server for any other replication contexts, delete the consumer server credentials entry with an objectclass of **ibm-slapdReplication**.
9. If there are any consumer server credentials entries with an objectclass of **ibm-slapdSupplier** that have only one **ibm-slapdReplicaSubtree** attribute value equal to the replication context that is being deleted, the entry can be deleted, or else, delete the **ibm-slapdReplicaSubtree** attribute value from the consumer server credentials entry.
10. Move the consumer server out of maintenance mode. See "Advanced replication maintenance mode" on page 536 for more information about advanced replication maintenance mode.

Once all replication topology entries are removed from the supplier and consumer servers, the **useAdvancedReplication** configuration option in the CDBM backend is set to **off** in both servers.

At this point, advanced replication is no longer configured between these two servers and each server is distinctly managing the data in the subtree that made up the replication context.

Advanced replication maintenance mode

Maintenance mode allows an LDAP root administrator to set up the server for advanced replication. This mode restricts access to all entries in an LDAP server. This allows the advanced replication topology to be fully configured on all servers participating in advanced replication.

While the LDAP server is in maintenance mode, an LDAP root administrator (for example, the **adminDN**) has unrestricted access to all entries in the server and can update operational attributes, such as **ibm-entryuuid** and **modifyTimestamp**, in the server. The operational attribute values in the **replicateOperationalAttributes** control are allowed to be updated when bound as an LDAP root administrator in maintenance mode. When the server is not in maintenance mode, an LDAP root administrator must specify the **Server Administration** control for the server to honor the operational attribute values in the **replicateOperationalAttributes** control. See Appendix C, "Supported server controls," on page 679 for more information about the **replicateOperationalAttributes** and **Server Administration** controls.

Consumer servers under maintenance mode continue to accept updates from supplier servers.

Pending replication entries are replicated to consumer servers, but, updates performed when in maintenance mode are not replicated.

Other users can bind to the LDAP server, but cannot access any entries within the server.

Specify the **-m** option on the server startup command to start the LDAP server in maintenance mode. The LDAP server MAINTMODE 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 operators console. If the command is entered from SDSF, it must be preceded by a slash (/).

```
f dssrv,maintmode state
```

where `state` can be **on** to turn on maintenance mode or **off** to turn maintenance mode off (and turn on normal mode).

Partial replication

Partial replication is an advanced replication feature that replicates only the specified entries and a subset of attributes for the specified entries within a subtree. The entries and attributes that are to be replicated are specified by an LDAP administrator with the appropriate authority. See Chapter 9, "Administrative

group and roles," on page 159 for more information about administrative role authority. Using partial replication, an administrator can enhance the replication bandwidth depending on the deployment requirements. With partial replication support, an LDAP administrator can allow entries that have certain object class values to be replicated to a consumer server. For example, entries that have an objectclass of **person** and only the **cn, sn, userPassword** attributes are allowed to be replicated but not the **description** attribute.

The replication filter entry can be specified in each individual replication agreement entry in the **ibm-replicationFilterDN** attribute value. See "Replication agreements" on page 498 for more information. A replication filter entry has a structural objectclass of **ibm-replicationFilter**. See Table 79 for the required attribute values for the **ibm-replicationFilter** objectclass. If an **ibm-replicationFilterDN** attribute value is not a valid replication filter entry or does not exist, replication for the replication agreement is suspended. If replication from the replication agreement is suspended, the replication filter entry must be added to the directory or the **ibm-replicationFilterDN** attribute value must be removed from the replication agreement entry. When replication is suspended for the replication agreement, it can be resumed by using the **Control replication** extended operation in the **ldapexop** utility. See "ldapexop utility" on page 255 for more information about the **ldapexop** utility.

The attributes that are to be replicated are specified using a replication filter in the **ibm-replicationFilterAttr** attribute. A set of attributes pertaining to an object class constitutes a replication filter. The list of attributes selected for an object class can either be a part of an inclusion list or an exclusion list. An inclusion list is list of attributes that are selected for replication while an exclusion list is list of attributes that are not selected for replication. See Table 79 for the required attribute values for the format of the **ibm-replicationFilterAttr** attribute values.

Table 79. **ibm-replicationFilter** *objectclass schema definition (required attributes)*

Attribute description and example
cn Common name of the replication filter entry. This attribute does not affect advanced replication configuration. Example: `cn: filter1`

Table 79. **ibm-replicationFilter** *objectclass schema definition (required attributes) (continued)*

Attribute description and example

ibm-replicationFilterAttr
 	
A multi-valued attribute that specifies a replication filter. A replication filter is based on the object class values of entries that are replicated. The filter can be an inclusion or exclusion filter.
 	
The following is the replication filter format:
`(objectclass=objclass):[!](attr1[,attr2]...)`
 	
where,
 	
`objclass` Specifies a valid objectclass in the servers schema. If an `*` is specified, then all other objectclasses not specified by other **ibm-replicationFilterAttr** attribute values (if any) in the replication filter entry, are subject to this replication filter.
 	
`!` If specified, indicates that the attribute type list is an exclusion list, otherwise, it is an inclusion list.
 	
`attr1, attr2` Specifies a list of valid attribute types in the servers schema. If an exclusion list, the attribute types in this list are not replicated for entries that have an objectclass value of `objclass`. If an inclusion list, the attribute types in this list are replicated for entries that have an objectclass value of `objclass`. An `*` can be specified to indicate all attribute types for the `objclass`.
 	
The following attributes are always replicated, irrespective of their presence in the exclusion list:
• Object class attributes of an entry
• Naming attribute
• All operational attributes (for example, **ibm-entryuuid** attribute values)
 	
Note:
1. If an attribute type is present in both an inclusion and an exclusion list, the exclusion takes precedence.
2. If there is not an **ibm-replicationFilterAttr** attribute value with `objclass` equal to `*`, no replication with entries that have an objectclass other than the ones explicitly specified is done. This acts as if an **ibm-replicationFilterAttr** attribute value of `(objectclass=*):!(*)` is specified on the replication filter entry.
 	
Example:
`ibm-replicationFilterAttr: (objectclass=person):(cn,sn)`

The example in Table 79 on page 537 allows the following replication to occur:

- Entries with an objectclass of **person** only have their **cn** and **sn** attribute values replicated.
- Entries with other objectclasses are not replicated.

Replication filter examples

The following are examples that explain the usages of replication filters:

Example 1

```
dn: cn=replicationfilter, cn=localhost
objectclass: ibm-replicationFilter
ibm-replicationFilterAttr: (objectclass=person):(*)
ibm-replicationFilterAttr: (objectclass=*): !(*)
```

The first **ibm-replicationFilterAttr** filter value indicates entries with an objectclass of **person** have all their attributes replicated. The second **ibm-replicationFilterAttr** filter value indicates entries with an objectclass other than **person** are not replicated. This means that only entries with an objectclass of **person** are replicated and no other entries are replicated.

Example 2

```
dn: cn=replicationfilter, cn=localhost
objectclass: ibm-replicationFilter
ibm-replicationFilterAttr: (objectclass=javaObject):(javaClassName,description)
ibm-replicationFilterAttr: (objectclass=javaNamingReference):(javaReferenceAddress)
ibm-replicationFilterAttr: (objectclass=*): !(javaReferenceAddress)
```

The first **ibm-replicationFilterAttr** filter value indicates entries with an objectclass of **javaObject** has their **javaClassName** and **description** attributes replicated. The second **ibm-replicationFilterAttr** filter value indicates entries with an objectclass of **javaNamingReference** has the **javaReferenceAddress** attribute replicated. The third **ibm-replicationFilterAttr** filter value indicates that the **javaReferenceAddress** attribute is not replicated for any entries other than **javaObject** and **javaNamingReference**.

Therefore, if an entry has an objectclass of **javaObject**, the **javaClassName** and **description** attributes are replicated. If an entry has an objectclass of **javaObject** and an auxiliary objectclass of **javaNamingReference**, the **javaClassName**, description, and **javaReferenceAddress** attributes are replicated. If an entry has an objectclass other than **javaObject** or **javaNamingReference**, all attributes except **javaReferenceAddress** are replicated.

Example 3

```
dn: cn=replicationfilter, cn=localhost
objectclass: ibm-replicationFilter
ibm-replicationFilterAttr: (objectclass=person):(cn,sn,userPassword)
ibm-replicationFilterAttr: (objectclass=inetOrgPerson):!(userPassword,employeeNumber)
ibm-replicationFilterAttr: (objectclass=*): !(*)
```

The first **ibm-replicationFilterAttr** filter value indicates entries with an objectclass of **person** have their **cn, sn**, and **userPassword** attribute values replicated. The second **ibm-replicationFilterAttr** filter value indicates that entries with an objectclass of **inetOrgPerson** do not have their **userPassword** and **employeeNumber** attribute values replicated. The third **ibm-replicationFilterAttr** filter value indicates that no other attributes are replicated if the objectclass is something other than **person** or **inetOrgPerson**.

Therefore, if an entry has an objectclass of **person**, the attributes **cn, sn**, and **userPassword** are replicated. If an entry has objectclasses of **person** and **inetOrgPerson**, only the **cn** and **sn** attributes are replicated. Because the **userPassword** attribute is present in both the inclusion and exclusion list, the **userPassword** attribute is eliminated because exclusion takes precedence over inclusion. If any other entry has an objectclass other than **person** or **inetOrgPerson**, no attributes are replicated.

SSL/TLS and advanced replication

SSL/TLS can be used to communicate between a replicating server (supplier, gateway, forwarder, or peer) and a replica server (consumer, gateway, forwarder, or peer).

Replica server with SSL/TLS enablement

Set up the replica server for SSL/TLS by updating the LDAP server configuration file if it is not already configured for SSL/TLS. An LDAP URL with a prefix of `ldaps://` is required in the **listen** configuration option in the replica server so that a secure connection can be configured. See “Setting up for SSL/TLS” on page 68 for more information.

If a SASL EXTERNAL bind is performed between the replicating and replica servers, the replica server must be configured to use server and client authentication. The **sslAuth** configuration option must be set to **serverClientAuth**. The replica server must have the replicating servers certificate in its key database file, RACF key ring, or PKCS #11 token. See "Setting up for SSL/TLS" on page 68 for more information.

Replicating server with SSL/TLS enablement

The replicating server acts as an SSL/TLS client to the replica server. To set up the replicating server to use simple binds, you must:

1. Run the **gskkyman** utility or the **RACDCERT** command as if you were the client. For more information, see *z/OS Cryptographic Services System SSL Programming* for the **gskkyman** utility or *z/OS Security Server RACF Command Language Reference* for the **RACDCERT** command. The key database file, RACF key ring, or PKCS #11 token must contain the replicating servers key pair and certificate. Receive the replicas self-signed certificate and mark it as trusted.
2. In the LDAP server configuration file on the replicating server:
   - Set the **sslKeyRingFile** configuration option to the replica key database file, RACF keyring, or PKCS #11 token that is created above.
   - If a key database file is used, set **sslKeyRingFilePW** to the password for the key database file, or set **sslKeyRingPWStashFile** to the file name where the password is stashed.
3. Ensure any environment variables that control SSL/TLS settings are properly defined in the LDAP server environment variable file. The environment variables for enabling TLS protocol levels are shared with the server definitions. For example, GSK_PROTOCOL_TLSV1_2=ON enables this protocol level for both inbound client connections to the replicating server and for outbound connections from the replicating server to the replica. However, since the replicating server acts as an SSL/TLS client to the replica, the environment variable usage for controlling cipher suites is as described for the client API. The SSL cipher format that you should use on the outbound connections to the replicas is controlled by the LDAP_SSL_CIPHER_FORMAT environment variable and then either the GSK_V3_CIPHER_SPECS or GSK_V3_CIPHER_SPECS_EXPANDED environment variable, depending on which format is chosen. The SSL cipher suites on the inbound client connections are controlled by the configured setting of sslCipherSpecs and can potentially share the setting that is specified on GSK_V3_CIPHER_SPECS_EXPANDED. Where settings are shared for both inbound client connections and outbound connections to replicas, the cipher list must include the necessary cipher suites for both sets of connections.
4. The **ibm-replicaURL** attribute value in the replication agreement entry must use an LDAP URL with a prefix of `ldaps://`. This indicates that an SSL connection is used between the replicating and replica servers. See Table 70 on page 498 information about the **ibm-replicaURL** attribute value.

The above procedure can also be used to set up the replicating server to use SASL EXTERNAL binds. The SSL-related configuration options in the LDAP server configuration file, if specified, represent the default values for the related optional attributes in **ibm-replicationCredentialsExternal** objects. These defaults can be overridden by specifying the optional attributes. See Table 73 on page 502 for more information about SASL EXTERNAL credentials entries.

Because the replicating server acts as an SSL/TLS client to the replica server, the replicating server binds with the replica server.

Displaying advanced replication configuration

Because many replication topologies can be configured in the LDAP server at one time, the LDAP server **DISPLAY REPLICAS** operator modify command is used to allow an LDAP administrator to easily query the current state of all configured replication contexts and replication agreements in the LDAP server. This command allows an LDAP administrator to verify that the advanced replication topologies are configured properly and are working as expected. See “Displaying performance information and server settings” on page 184 for more information about the LDAP server **DISPLAY REPLICAS** operator modify command.

If the LDAP server **DISPLAY REPLICAS** operator modify command output indicates that there are problems in a replication agreement entry, LDAP search requests can be used to query the replication agreement entry's operational attributes to determine the replication problems. See “Monitoring and diagnosing advanced replication problems” on page 543 for more information.

Command line tasks for managing replication

This topic contains information about the use of the tools that are at your disposal to check the current advanced replication status for each of your configured replication agreements. These procedures can be used to recover from out of sync conditions between servers participating in advanced replication.

Advanced replication related extended operations

A set of extended operations is provided to allow administrators the opportunity to manage all aspects of advanced replication. Specifically, the extended operations allow an administrator to do the following:

- Propagate any replication topology entries to all consumers
- Manage the replication queue, especially in the context of replication scheduling
- Manage any replication-related errors
- Manage the quiesce state of the replication context
- Suspend or resume replication processing

The extended operations also allow for an administrator to manage a specific server and then have the server cascade the management operation to any of its consumers. These extended operations require the user to bind as an LDAP root administrator or an administrator with the appropriate authority. See “Administrative roles and extended operations” on page 168 for more information about administrative role authority to run these extended operations. The **ldapexop** utility is provided to invoke the advanced replication extended operations. This utility provides a command-line interface to all the advanced replication extended operations. See “ldapexop utility” on page 255 for more information.

Table 80 on page 542 summarizes the extended operations with their **ldapexop** operation value.

Note: The term, **cascade**, is used to describe the process of a supplier server issuing an extended operation it processed to one or all of its consumers.

Table 80. Description of advanced replication extended operations on the **ldapexop** *utility*

ldapexop operation	**ldapexop description**	**Overview**
cascrepl	**Cascading control replication** extended operation	This extended operation can quiesce, unquiesce, or force immediate replication of all pending changes (even if scheduling is dictated another way). When the extended operation is performed on the supplier that this extended operation was issued against, it proceeds to cascade the extended operation to one or all of its consumers. If forcing immediate replication of all pending changes, two variants of the extended operation can be issued. If the replication of all pending changes must complete before the cascade step, an administrator can use the **wait** option. If there are no dependencies on the completion of the replication operations before cascading, the **replnow** option can be used. See "Cascading control replication" on page 696 for more information about the **Cascading control replication** extended operation.
controlqueue	**Control replication queue** extended operation	This extended operation can skip one or all pending changes in the advanced replication queue. See "Control replication queue" on page 703 for more information about the **Control replication queue** extended operation.
controlrepl	**Control replication** extended operation	This extended operation suspends or resumes all advanced replication-related activity. Also, given any replication schedule objects that might exist, resuming replication does not necessarily cause the immediate replication of any pending changes. Instead, in addition to resume, this extended operation can be used to begin the immediate replication of any pending changes, even if replication schedule objects have deferred replication. See "Control replication" on page 700 for more information about the **Control replication** extended operation.
controlreplerr	**Control replication error log** extended operation	This extended operation can delete, retry, or show, any replication operations that resulted in an unsuccessful return code returned to the supplier from the consumer. The show options returns an LDIF representation of the unsuccessful replication operation. See "Control replication error log" on page 702 for more information about the **Control replication error log** extended operation.
quiesce	**Quiesce or unquiesce context** extended operation	This extended operation can quiesce or unquiesce a replication context. A quiesced replication context typically cannot accept any LDAP update operations. To perform LDAP update operations on a quiesced context, the update operations must be done by an LDAP administrator with the appropriate authority and using the **Server Administration** control. See Chapter 9, "Administrative group and roles," on page 159 for more information about administrative role authority. See "Quiesce or unquiesce context" on page 711 for more information about the **Quiesce or unquiesce context** extended operation.

Table 80. Description of advanced replication extended operations on the **ldapexop** *utility (continued)*

ldapexop operation	ldapexop description	Overview
repltopology	**Replication topology** extended operation	This extended operation propagates advanced replication topology-related entries to all consumers. It then cascades this extended operation to all the consumers. This results in a cascading of the topology-related entries to all servers that participate in replication for a given replication context. See "Replication topology" on page 714 for more information about the **Replication topology** extended operation.

Viewing replication configuration information

A great deal of information related to replication activity is available using searches. To see the replication topology information related to a particular replicated subtree, you can do a subtree search with the base set to the DN of the subtree and the filter set as (**objectclass=ibm-repl***) to find the subentry that is the base of the topology information.

```
ldapsearch -p port -D adminDN -w adminPW -b baseDn objectclass=ibm-repl*
```

The objects returned includes the replica group itself, plus the following:

- Entries with an objectclass of **ibm-replicaSubentry** for each server that replicates data within this replication context. These entries are replica subentries that contain a server ID attribute and the role that servers plays in the replication topology. See "Replica subentries" on page 497 for more information about replica subentries.
- For each replica subentry, there is a replication agreement entry for each consumer server that receives replication updates from the server described by the replica subentry. See "Replication agreements" on page 498 for more information about replication agreement entries.

Monitoring and diagnosing advanced replication problems

An LDAP administrator can monitor the state of advanced replication processing and troubleshoot problems by using LDAP search requests to retrieve operational attributes available for the roots of the replication contexts (entries with an objectclass of **ibm-replicationContext**) and replication agreements (entries with an objectclass of **ibm-replicationAgreement**). Because these are operational attributes, either the + attribute or each individual attribute must be requested on a search request in order to be returned. Also, operational attributes cannot be used in search filters.

The following tables describe the operational attributes for the replication context and replication agreement entries. Replication context entries use the auxiliary objectclass of **ibm-replicationContext** and replication agreement entries use the structural objectclass **ibm-replicationAgreement**. See Table 81 on page 544 for the operational attributes for the **ibm-replicationContext** objectclass. See Table 82 on page 544 for the operational attributes for the **ibm-replicationAgreement** objectclass.

When retrieved for a replication context or replication agreement entry, the operational attributes provide information concerning that entry. It is important to take notice of attributes that have values that contain *failureId* or *changeId* values. The *failureId* and *changeId* numbers increase sequentially. However, some numbers might be skipped by the server for various reasons. For example, if DB2 is

restarted while the server is running, the *changeId* might skip numbers. These IDs are often required when working with the **Control replication error log** and the **Control replication queue** extended operations with the **ldapexop** utility. See "ldapexop utility" on page 255 for more information about the **ldapexop** utility.

Table 81. **ibm-replicationContext** *operational attributes*

Attribute and description
ibm-replicationThisSeverIsMaster A boolean (true or false) indicating whether the server is the master of the replication context. If set to true, the server is the master of the replication context. If set to false, the server is a not the master of the replication context.
ibm-replicationIsQuiesced A boolean (true or false) indicating whether the replication context is quiesced. If set to true, the replication context is quiesced. If set to false, the replication context is not quiesced. Updates under a quiesced replication context are restricted to an LDAP root administrator if using the **Server Administration** control (OID 1.3.18.0.2.10.15), and any replication master DNs with authority under this context. Advanced replication continues for a quiesced context. If the server is restarted, all replication contexts are then unquiesced.

See Table 67 on page 496 for the optional non-operational attribute for the **ibm-replicationContext** objectclass.

Table 82. **ibm-replicationAgreement** *operational attributes*

Attribute and description
ibm-replicationChangeLDIF The LDIF representation of the next pending change that has not yet been replicated and has resulted in advanced replication being stalled to the consumer server. If there is not a stalled replication change, the value is **N/A**. Examples of when an advanced replication queue might be stalled include: 1. A replication change failed because of an **LDAP_TIMEOUT** return code. 2. The backend replication table has reached the maximum number of errors allowed on the supplier server within this backend while attempting to replicate a change to a consumer server. See Table 11 on page 141 for more information about the **ibm-slapdReplMaxErrors** attribute value.
ibm-replicationFailedChangeCount Specifies the number of advanced replication operations that have failed in this replication agreement. This number is shared among all replication agreement entries on the backend level by the **ibm-slapdReplMaxErrors** attribute in the CDBM backend configuration entry **cn=Replication, cn=Configuration**. See Table 11 on page 141 for more information about the **ibm-slapdReplMaxErrors** attribute value.

Table 82. **ibm-replicationAgreement** *operational attributes (continued)*

Attribute and description

ibm-replicationFailedChanges
 	
A multi-valued attribute that lists all the logged replication operations that have failed. The number of attribute values is shared among all replication agreement entries on the backend level by the **ibm-slapdReplMaxErrors** attribute in the CDBM backend configuration entry **cn=Replication, cn=Configuration**. See Table 11 on page 141 for more information about the **ibm-slapdReplMaxErrors** attribute value.
 	
A string value of the form: ***failureId timestamp returnCode numOfAttempts changeId operation entryDn***
 	
The *failureId* identifies the update that has failed to replicate to the consumer server. The *failureId* is used with the **Control replication error log** extended operation to display, delete, or retry the failing replication update. The **ldapexop** utility supports the **Control replication error log** extended operation. See "ldapexop utility" on page 255 for more information about the **ldapexop** utility.
 	
The *timestamp* is the time in Zulu format when this operation was last attempted to be replicated to the consumer server.
 	
The *returnCode* is the LDAP return code from the consumer server.
 	
The *numOfAttempts* is the number of times the error has been tried again on the consumer server.
 	
The *changeId* is the ID that this *failureId* had when it was in the pending replication queue.
 	
The *operation* indicates the update operation that encountered the failure. It has one of the following values: **add, delete, modify**, or **modifydn**
 	
The *entryDn* indicates the distinguished name of the entry that caused the failure.
 	
Example:
`ibm-replicationfailedchanges: 1 20050407202221Z 68 1 170814 add cn=entry-85,o=IBM,c=US`
 	
`failureId: 1`
`timestamp: 20050407202221Z`
`returnCode: 68`
`numOfAttempts: 1`
`changeId: 170814`
`operation: add`
`entryDn: cn=entry-85,o=IBM,c=US`

ibm-replicationLastActivationTime Specifies the Zulu format timestamp when advanced replication actively began replicating queued updates.
ibm-replicationLastChangeID Specifies the replication change ID of the last successfully completed advanced replication update.
ibm-replicationLastFinishTime Specifies the Zulu format timestamp when advanced replication updates in the queue were all attempted and the server awaits a new scheduled start time or more operations to appear in the advanced replication queue. See "Schedule entries" on page 503 for more information about replication schedule entries.

Table 82. **ibm-replicationAgreement** *operational attributes (continued)*

Attribute and description

ibm-replicationLastResult
 	
A description of the result from the last advanced replication operation or connection attempt to a consumer server.
 	
A string value of the form: ***timestamp changeId returnCode operation entryDn***
 	
The *timestamp* is the time in Zulu format when this operation was last attempted to be replicated to the consumer server.
 	
The *changeId* is the ID of the last replication update.
 	
The *returnCode* is the LDAP return code from the consumer server.
 	
The *operation* indicates the last LDAP operation. It has one of the following values: **add, connect, delete, modify**, or **modifydn**
 	
The *entryDn* indicates the distinguished name of the entry that was last added, deleted, modified, or renamed. If *operation* is **connect**, *entryDn* is set to **NULL**.
 	
Example:
`ibm-replicationLastResult: 20050412140436Z 19 81 add cn=testpendingchange,o=ibm,c=us`
 	
`timestamp: 20050412140436Z`
`changeId: 19`
`returnCode: 81`
`operation: add`
`entryDn: cn=testpendingchange,o=ibm,c=us`

ibm-replicationLastResultAdditional The descriptive reason code message text that supplements the return code message with the purpose of providing additional information from the last replication attempt.
ibm-replicationNextTime Specifies the Zulu format timestamp of the next time advanced replication would begin if pending changes existed. When this value is set to **19000101000000z**, replication begins immediately when a change is ready to be replicated if the **ibm-replicationState** operational attribute is set to **active**.
ibm-replicationPendingChangeCount The number of replication operations that are waiting to be replicated to a consumer server.

ibm-replicationPendingChanges
 	
A multi-valued attribute that lists all changes waiting to be replication to a consumer server.
 	
A string value of the form: ***changeId operation entryDn***
 	
The *changeId* is the ID of the pending replication update.
 	
The *operation* indicates the LDAP operation that is pending. It has one of the following values: **add, delete, modify**, or **modifydn**
 	
The *entryDn* indicates the distinguished name of the entry that is to be added, deleted, modified, or renamed.
 	
Example:
`ibm-replicationpendingchanges: 19 add cn=test1,o=ibm,c=us`
 	
`changeId: 19`
`operation: add`
`entryDn: cn=test1,o=ibm,c=us`

Table 82. **ibm-replicationAgreement** *operational attributes (continued)*

Attribute and description
ibm-replicationState Identifies the current state of the advanced replication queue. It has one of the following values: • **active** - Indicates that advanced replication is occurring from this replication agreement. • **binding** - Indicates that the replication agreement is in the process of authenticating with the consumer server. • **connecting** - Indicates that the replication agreement is attempting to contact the consumer server. • **on hold** - Indicates that the replication agreement is on hold. Replication updates to the consumer server are queued until the replication agreement is resumed. • **ready** - Indicates immediate replication mode, ready to send updates as they occur. • **retrying** - Indicates that the server retries the current change every 60 seconds until it succeeds. The retrying state occurs when a consumer server is restarted, the replication backend table is full, the current replicated update is failing, or when there is an **LDAP_TIMEOUT** return code from the consumer server. Retrying is a likely symptom that advanced replication might be stalled and LDAP administrator intervention is required to get it running again. See "Recovering from advanced replication errors" for the steps on how to recover from out of sync conditions between supplier and consumer servers. • **suspended** - Indicates that the replication agreement is suspended. No additional replication updates are sent to the consumer server by this agreement (until it returns to the ready state). • **waiting** - Indicates that the replication agreement is currently waiting for the next scheduled replication to occur. See "Schedule entries" on page 503 for more information about replication schedule entries.

See Table 70 on page 498 for the required non-operational attributes for the **ibm-replicationAgreement** objectclass. See Table 71 on page 499 for the optional non-operational attributes for the **ibm-replicationAgreement** objectclass.

Recovering from advanced replication errors

Replication errors can be handled proactively, before they are allowed to accumulate, or reactively, after replication has already stalled. Replication stalls occur when the number of failures reaches the limit as specified by the **ibm-slapdReplMaxErrors** attribute value in the **cn=Replication,cn=configuration** entry. See Table 11 on page 141 for more information about the **cn=Replication,cn=configuration** entry.

When replication is stalled, the latest failed change occupies the beginning of the pending changes queue. The latest failed change gets retried every minute until it succeeds or the failed change is removed from the queue by an LDAP administrator with the appropriate authority. See Chapter 9, "Administrative group and roles," on page 159 for more information about administrative role authority. When this failed change occupies the lead position in the pending replication queue, all other replication updates are blocked and replication is stalled.

The options for handling stalled replication are:

1. Increase the size of the **ibm-slapdReplMaxErrors** attribute in the **cn=Replication,cn=configuration** entry. This allows more replication failures to be stored in the backend where the replication agreement entry exists.
2. Delete or retry one or more failed replication changes.
3. Skip the latest failed replication change.
4. If the stalled replication problem is severe enough, the entire replication context where the replication agreement entry exists might need to be resynchronized. In order to do this, you must:
   a. Quiesce the replication context

b. Suspend replication for all replication agreements
c. Delete all failed replication changes for all replication agreements
d. Skip all pending changes for all replication agreements
e. Resynchronize the replication context
f. Resume replication for the suspended replication agreements
g. Unquiesce the replication context

The following operational attributes in the replication agreement entry can be queried to determine what to do:

1. The **ibm-replicationChangeLdif** operational attribute in the replication agreement entry shows the LDIF representation of the latest failure. The **ibm-replicationLastResult** and **ibm-replicationLastResultAdditional** operational attributes in the replication agreement have further detail for the reason the change failed.
2. The **ibm-replicationPendingChanges** operational attribute in the replication agreement shows the change ID, the operation type, and the target DN of the next changes to be replicated. The number of pending changes that are displayed is limited by the **ibm-slapdMaxPendingChangesDisplayed** attribute in the **cn=Replication,cn=configuration** entry. See Table 11 on page 141 for more information about the **ibm-slapdMaxPendingChangesDisplayed** attribute. See Table 82 on page 544 for more information about the **ibm-replicationPendingChanges** operational attribute.
3. The **ibm-replicationFailedChanges** operational attribute in the replication agreement shows each of the failed changes, including the failure ID. See Table 82 on page 544 for more information about the **ibm-replicationFailedChanges** operational attribute.
4. The **Control replication error log** extended operation can be used to display information about a failure by providing the *failureId* obtained from the **ibm-replicationFailedChanges** operational attribute. The **controlreplerr** extended operation **-show** option in the **ldapexop** utility can be used to display the latest failure. See "ldapexop utility" on page 255 for more information about the **ldapexop** utility.

When the latest and all previous failures are understood, an LDAP administrator must decide whether to fix the replication failures individually or resynchronize the entire replication context. The options are:

1. Increase the size of the **ibm-slapdReplMaxErrors** attribute in the **cn=Replication,cn=configuration** entry. This allows more replication failures to be stored in the backend where the replication agreement entry exists. See Table 11 on page 141 for more information about the **ibm-slapdReplMaxErrors** attribute.
2. Delete or retry one or more failed changes for the replication agreement by using the **Control replication error log** extended operation with the **ldapexop** utility. The **-retry** option on the **controlreplerr** extended operation in the **ldapexop** utility allows a single failure (identified by its *failureId*) to be retried or all failures to be retried. The ability to retry all failures is especially useful when you have corrected the problem that caused a change to fail the first time. When a failed change is retried successfully, it is removed from the list of failed changes and there is space for a new one. The **-delete** option on the **controlrepler** extended operation in the **ldapexop** utility allows a single failure (identified by its *failureId*) to be deleted or all failures to be deleted. This delete option is especially useful when a change is deemed to be unnecessary, the problem has been fixed manually, or a synchronization tool such as the

ldapdiff utility has been used to resynchronize the directories. Deleting a failed change frees space in the list of failed changes so that a new failure can be added. See "ldapexop utility" on page 255 for more information about the **ldapexop** utility. See "ldapdiff utility" on page 247 for more information about the **ldapdiff** utility.

3. Skip the latest failure for the replication agreement by using the **Control replication queue** extended operation. The **ldapexop** utility supports the **Control replication queue** extended operation that allows the next pending change (identified by its *changeId*) or all pending changes to be skipped. This extended operation is useful when the **ibm-slapdReplMaxErrors** attribute in the **cn=Replication,cn=configuration** entry is set to 0 in which case the replication failure is not allowed and replication stalls on the first failure. Also, the **Control replication queue** extended operation is useful when replication failures are not deleted, the **ibm-slapdReplMaxErrors** attribute value is increased, or after using the **ldapdiff** utility to resynchronize the replication context. See "ldapexop utility" on page 255 for more information about the **ldapexop** utility.
4. If there are multiple failed and pending replication changes, the entire replication context where the replication agreement entry exists might need to be resynchronized. In order to do this, you must:
   a. Quiesce the replication context on all servers in the replication topology by using the **Cascading control replication** extended operation on the **ldapexop** utility. The **Cascading control replication** extended operation is targeted against the master server which in turn quiesces the replication context on all consumer servers. A quiesced replication context only accepts updates from an LDAP root administrator when using the **Server Administration** control and any replication master server DNs with authority under this context. See "Cascading control replication" on page 696 for more information about the **Cascading control replication** extended operation. See "ldapexop utility" on page 255 for more information about the **ldapexop** utility.
   b. Suspend replication for all replication agreements in the replication context by using the **Control replication** extended operation on the **ldapexop** utility. A suspended replication agreement queues all replication changes updates until it is resumed. See "Control replication" on page 700 for more information about the **Control replication** extended operation.
   c. Use the **ldapdiff** utility with the **-L** option to compare the replication contexts on each of the servers within the replication context. The **-L** option allows the entry differences to be written to an output LDIF file. See "ldapdiff utility" on page 247 for more information about the **ldapdiff** utility.
   d. Delete all failed replication changes for all replication agreements by using the **Control replication error log** extended operation on the **ldapexop** utility. See "Control replication error log" on page 702 for more information about the **Control replication error log** extended operation.
   e. Skip all pending replication changes by using the **Control replication queue** extended operation on the **ldapexop** utility. See "Control replication queue" on page 703 for more information about the **Control replication queue** extended operation.
   f. Resynchronize the replication context by using the fix option on the **ldapdiff** utility.
   g. Resume replication for all suspended replication agreements by using the **Control replication** extended operation on the **ldapexop** utility.

h. Unquiesce the replication context on all servers in the replication topology by using the **Cascading control replication** extended operation on the **ldapexop** utility.

The other methodology for handling replication failures is to take a proactive, preventive approach. An LDAP administrator monitors the replication failure queue and resolves problems before the queue reaches capacity and replication stalls. An LDAP administrator with the appropriate authority can use the **Control replication error log** extended operation and the **ibm-replicationFailedChanges** and **ibm-replicationState** operational attributes in the replication agreement entry to monitor the current replication status. See Chapter 9, "Administrative group and roles," on page 159 for information about administrative authority.

Advanced replication error recovery example

This advanced replication error recovery example uses the master-replica topology that has been configured in "Creating a master-replica topology" on page 515. This example assumes the **ibm-slapdReplMaxErrors** attribute value in the **cn=Replication,cn=configuration** entry is set to one.

An LDAP administrator periodically monitors the replication status of the replication agreement in the `o=ibm,c=us` replication context by querying the replication agreement operational attribute values. See Table 82 on page 544 for more information about the replication agreement operational attributes.

Note: Operational attributes are only returned on search requests when either the + attribute is specified or each operational attribute is requested.

The current replication status from the master to the replica can be determined by using the following **ldapsearch** command to retrieve the replication agreement entry. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information about the **ldapsearch** utility.

```
ldapsearch -p 389 –h server1.us.ibm.com -D adminDn -w adminPw -b o=ibm,c=us
 "(objectclass=ibm-replicationAgreement)" "*" ibm-replicationChangeLdif
 ibm-replicationFailedChangeCount ibm-replicationFailedChanges ibm-replicationLastActivationTime
 ibm-replicationLastChangeID ibm-replicationLastFinishTime ibm-replicationLastResult
 ibm-replicationLastResultAdditional ibm-replicationNextTime ibm-replicationPendingChangeCount
 ibm-replicationPendingChanges ibm-replicationState
```

The **ldapsearch** command returns the following entry:

```
cn=Replica, ibm-replicaServerId=Master, ibm-replicaGroup=default, o=ibm, c=us
objectclass=top
objectclass=ibm-replicationAgreement
ibm-replicaconsumerid=Replica
ibm-replicaurl=ldap://server1.us.ibm.com:389
ibm-replicacredentialsdn=cn=ReplicaBindCredentials,o=ibm, c=us
description=Replication agreement from master to replica
cn=Replica
ibm-replicationonhold=FALSE
ibm-replicationstate=retrying
ibm-replicationpendingchanges=46 modify OU=SUB,O=IBM,C=US
ibm-replicationpendingchangecount=1
ibm-replicationnexttime=19000101000000
ibm-replicationlastresultadditional=R004071 DN 'OU=SUB,O=IBM,C=US' does not exist
 (ldbm_process_request:406)
ibm-replicationlastresult=20090206145054Z 46 32 modify OU=SUB,O=IBM,C=US
ibm-replicationlastfinishtime=20090206144954Z
ibm-replicationlastchangeid=45
ibm-replicationlastactivationtime=20090206144354Z
ibm-replicationfailedchanges=12 20090206144954Z 32 1 45 add cn=entry,ou=sub,o=ibm,c=us
ibm-replicationfailedchangecount=1
ibm-replicationchangeldif=
dn: ou=sub,o=ibm,c=us
control: 2.16.840.1.113730.3.4.2 true
control: 1.3.18.0.2.10.19 false:: MIGPMCAKAQIwGwQNbW9kaWZpZXJzTmFtZTEKBAhjbj1
```

```
 hZG1pbjAwCgECMCsED21vZGlmeVRpbWVzdGFtcDEYBBYyMDA5MDIwNjE0NDg1My41ODM4MjVaMDk
 KAQIwNAQYUmVwbGljYXRpb25CYXNlVGltZXN0YW1wMRgEFjIwMDkwMjA2MTM0ODQ2Ljc4Njg4NFo
 =
changetype: modify
add: description
description: A small division
```

The following analysis of the replication agreement entry can be performed:

1. The **ibm-replicationState** operational attribute value is set to `retrying` which indicates replication is currently stalled. Replication is stalled because the number of replication failures exceeds one. (The **ibm-slapdMaxReplErrors** attribute value has been set to one in the **cn=Replication,cn=configuration** entry).
2. The **ibm-replicationChangeLdif** operational attribute in the replication agreement shows the LDIF representation of the latest failure. The LDIF shows that the last failure is a modify of the `ou=sub,o=ibm,c=us` entry on the consumer server. The **ibm-replicationLastResult** and **ibm-replicationLastResultAdditional** operational attributes in the replication agreement indicate that the modify failed on the consumer server because the `ou=sub,o=ibm,c=us` entry does not exist.
3. The **ibm-replicationPendingChanges** operational attribute in the replication agreement shows the *changeId* of the next pending update is 46. The next pending change is also the same modify operation of the `ou=sub,o=ibm,c=us` entry. It will be replicated to the consumer server after the add failure in the **ibm-replicationFailedChanges** operational attribute is resolved.
4. The **ibm-replicationFailedChanges** operational attribute in the replication agreement shows one failed replication update. The attribute value indicates that the *failureId* is 12, the LDAP return code from the consumer server is 32, it is an add operation of the `cn=entry,ou=sub,o=ibm,c=us` entry, and the supplier server has tried once to replicate the update.

To determine why the addition of the `cn=entry,ou=sub,o=ibm,c=us` entry failed, the **ldapexop** utility can be used to perform a **Control replication error log** extended operation to show the failed replication change. See "ldapexop utility" on page 255 for more information about the **ldapexop** utility.

The following **ldapexop** command can be used to show the LDIF representation of failed replication change that has a *failureId* of 12.

```
ldapexop -p 389 –h server1.us.ibm.com -D adminDn -w adminPw -op controlreplerr
 -ra "cn=Replica, ibm-replicaServerId=Master, ibm-replicaGroup=default,
 o=ibm,c=us" -show 12
```

The **ldapexop** command returns the following:

```
dn: cn=entry,ou=sub,o=ibm,c=us
control: 2.16.840.1.113730.3.4.2 true
control: 1.3.18.0.2.10.19 false:: MIGnMDAKAQAwKwQPbW9kaWZ5dGltZXN0YW1wMRgEFjI
   wMDkwMjA2MTQ0MzU0LjY1NzcwMFowIAoBADAbBA1tb2RpZmllcnNuYW1lMQoECGNuPWFkbWluMDA
   KAQAwKwQPY3JlYXRldGltZXN0YW1wMRgEFjIwMDkwMjA2MTQ0MzU0LjY1NzcwMFowHwoBADAaBAx
   jcmVhdG9yc25hbWUxCgQIY249YWRtaW4=
changetype: add
cn: entry
ibm-entryuuid: A091A000-4CAA-198C-8D7D-402084027431
sn: entry
objectclass: person
objectclass: top
```

An LDAP administrator can either fix the replication differences manually or use the **ldapdiff** utility to resynchronize the replication contexts on all servers in the replication topology. The **ldapdiff** utility is a useful tool for comparing and verifying that the entries within a replication context on supplier and consumer

server are synchronized. For the purposes of this example, an LDAP administrator has chosen to resynchronize the replication context by using the **ldapdiff** utility. See "ldapdiff utility" on page 247 for more information about the **ldapdiff** utility.

Before you use the **ldapdiff** utility to compare or fix entries within a replication context, quiesce the replication context on all servers within the replication topology by using the **Cascading control replication** extended operation quiesce option on the **ldapexop** utility. See "ldapexop utility" on page 255 for more information about the **ldapexop** utility.

The following **ldapexop** command quiesces the `o=ibm,c=us` replication context on the master and replica server in the replication topology:

```
ldapexop –p 389 –h server1.us.ibm.com –D adminDn –w adminPw –op cascrepl –action
 quiesce –rc "o=ibm,c=us"
```

After the replication context is quiesced on all servers, the **Control replication** extended operation can be used to suspend replication for all replication agreements within the replication context.

The following **ldapexop** command suspends replication for all replication agreements in the replication context `o=ibm,c=us`. The `cn=Replica, ibm-replicaServerId=Master, ibm-replicaGroup=default, o=ibm, c=us` is the only replication agreement within the `o=ibm,c=us` replication context so that it is the only agreement that is suspended.

```
ldapexop –p 389 –h server1.us.ibm.com –D adminDn –w adminPw –op controlrepl
 –action suspend –rc "o=ibm,c=us"
```

The following **ldapdiff** command is run to compare the entries within the replication context `o=ibm,c=us` on the master server on `server1.us.ibm.com` and the replica server on `server2.us.ibm.com`. If there are any differences between the two servers, they are written to the output LDIF file called `differences.ldif`. The **ldapdiff -a** option is specified to write the **Server Administration** control to the output LDIF for each entry that is different between the two servers. See "Server Administration" on page 690 for more information about the **Server Administration** control.

```
ldapdiff –a -b "o=ibm,c=us" -L differences.ldif -sh server1.us.ibm.com -sp 389
 -sD adminDn -sw adminPw
 -ch server2.us.ibm.com -cp 389 -cD adminDn –cw adminPw
```

where `differences.ldif` contains:

```
dn: ou=sub,o=ibm,c=us
control: 1.3.18.0.2.10.15 true
control: 1.3.18.0.2.10.19 false::
 MIGnMB8KAQAwGgQMY3JlYXRvcnNOYW1lMQoECGNuPWFkbWluMDAKAQAwKQP
 Y3JlYXRlVGltZVN0YW1wMRgEFjIwMDkwMjA2MTM0ODQ2Ljc4Njg4NFowIAoB
 ADAbBA1tb2RpZmllcnNOYW1lMQoECGNuPWFkbWluMDAKAQAwKQPbW9kaWZ5
 VGltZVN0YW1wMRgEFjIwMDkwMjA2MTQ0ODUzLjU4MzgyNVo=
changeType: add
ibm-entryuuid: C01B9000-3FBE-198C-98A7-402084027431
ou: sub
description: A small division
objectclass: organizationalUnit
objectclass: top

dn: cn=entry,ou=sub,o=ibm,c=us
control: 1.3.18.0.2.10.15 true
control: 1.3.18.0.2.10.19 false::
 MIGnMB8KAQAwGgQMY3JlYXRvcnNOYW1lMQoECGNuPWFkbWluMDAKAQAwKQP
 Y3JlYXRlVGltZVN0YW1wMRgEFjIwMDkwMjA2MTQ0MzU0LjY1NzcwMFowIAoB
 ADAbBA1tb2RpZmllcnNOYW1lMQoECGNuPWFkbWluMDAKAQAwKQPbW9kaWZ5
 VGltZVN0YW1wMRgEFjIwMDkwMjA2MTQ0MzU0LjY1NzcwMFo=
changeType: add
ibm-entryuuid: A091A000-4CAA-198C-8D7D-402084027431
```

```
objectclass: person
objectclass: top
sn: entry
cn: entry
```

The contents of the `differences.ldif` file indicates that the `ou=sub,o=ibm,c=us` entry does not exist on the consumer server. This explains why the addition of the child entry `cn=entry,ou=sub,o=ibm,c=us` failed on the consumer server.

Before synchronizing entries within a replication context on the master and replica servers, all replication failures are deleted and all pending replication changes are skipped. Replication failures are deleted by using the **Control replication error log** extended operation on the **ldapexop** utility. Pending replication changes are skipped by using the **Control replication queue** extended operation on the **ldapexop** utility.

The following **ldapexop** command deletes all failed replication failures from the backend replication table, `cn=Replica, ibm-replicaServerId=Master, ibm-replicaGroup=default, o=ibm, c=us`:

```
ldapexop -p 389 –h server1.us.ibm.com -D adminDn -w adminPw -op controlreplerr
 -delete all -ra "cn=Replica, ibm-replicaServerId=Master, ibm-replicaGroup=default,
 o=ibm, c=us"
```

The following **ldapexop** command skips (deletes) all pending replication changes from the replication queue:

```
ldapexop -p 389 –h server1.us.ibm.com -D adminDn -w adminPw -op controlqueue
 –skip all -ra "cn=Replica, ibm-replicaServerId=Master, ibm-replicaGroup=default,
 o=ibm, c=us"
```

To synchronize the `o=ibm,c=us` replication context on the master and replica servers, run the **ldapdiff** utility again with the **-F** (Fix) option specified or use the **ldapmodify** command to add the entries in the `differences.ldif` file to the consumer server.

Because the master and replica servers are now synchronized, the replication agreement can now be resumed and the replication context unquiesced. The replication agreement is resumed by using the **Control replication** extended operation on the **ldapexop** utility. The replication context is unquiesced on all servers in the replication topology by using the **Cascading control replication** extended operation on the **ldapexop** utility.

The following **ldapexop** command resumes replication for the replication agreement, `cn=Replica, ibm-replicaServerId=Master, ibm-replicaGroup=default, o=ibm, c=us`:

```
ldapexop –p 389 –h server1.us.ibm.com –D adminDn –w adminPw –op controlrepl
 –action resume –ra "cn=Replica, ibm-replicaServerId=Master, ibm-replicaGroup=default,
 o=ibm, c=us"
```

The following **ldapexop** command unquiesces the replication context `o=ibm,c=us` on all servers in the replication topology:

```
ldapexop –p 389 –h server1.us.ibm.com –D adminDn –w adminPw –op cascrepl
 –action unquiesce –rc "o=ibm,c=us"
```

The current replication status from the master to the replica can be determined by using the following **ldapsearch** command to retrieve the replication agreement entry:

```
ldapsearch -p 389 –h server1.us.ibm.com -D adminDn -w adminPw -b "o=ibm,c=us"
 "(objectclass=ibm-replicationAgreement)" "*" ibm-replicationChangeLdif
 ibm-replicationFailedChangeCount ibm-replicationFailedChanges ibm-replicationLastActivationTime
 ibm-replicationLastChangeID ibm-replicationLastFinishTime ibm-replicationLastResult
 ibm-replicationLastResultAdditional ibm-replicationNextTime ibm-replicationPendingChangeCount
 ibm-replicationPendingChanges ibm-replicationState
```

The **ldapsearch** command returns the following entry:

```
cn=Replica, ibm-replicaServerId=Master, ibm-replicaGroup=default, o=ibm, c=us
objectclass=top
objectclass=ibm-replicationAgreement
ibm-replicaconsumerid=Replica
ibm-replicaurl=ldap://server2.us.ibm.com:389
ibm-replicacredentialsdn=cn=ReplicaBindCredentials,o=ibm, c=us
description=Replication agreement from master to replica
cn=Replica
ibm-replicationonhold=FALSE
ibm-replicationstate=ready
ibm-replicationpendingchangecount=0
ibm-replicationnexttime=19000101000000
ibm-replicationlastfinishtime=20090206165454Z
ibm-replicationlastchangeid=46
ibm-replicationlastactivationtime=20090206144354Z
ibm-replicationfailedchangecount=0
ibm-replicationchangeldif=N/A
```

Because the **ibm-replicationState** operational attribute value in the replication agreement entry is set to ready, replication from the master to the replica is now no longer stalled.

Chapter 27. Alias

Alias support provides a means for a TDBM, LDBM, or CDBM directory entry to point to another entry in the same directory. An alias entry 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.

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 z/OS LDAP department:

```
"ou=LDAPZOS,o=IBM"
```

The alias entry points to the actual z/OS LDAP department:

```
"ou=DEPTC8NG,ou=Poughkeepsie,o=IBM_US,o=IBM"
```

This provides easier access to the entries of the z/OS LDAP developers, using public names such as:

```
"cn=kmorg,ou=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 always part of the LDAP server schema.

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

An alias is a directory entry containing either the **alias** structural object class or the **aliasObject** auxiliary object class. Both of these object classes require the **aliasedObjectName** attribute (the **aliasedEntryName** alternate name can also be used). The **extensibleObject** object class should also be specified if the **alias** object class is used in order to add the RDN attributes for the alias entry.

An alias entry must be a leaf entry. This means that no ancestor of an entry can be an alias entry. In addition, an alias entry cannot also be a referral entry. A suffix entry can be an alias entry. In this case, the suffix will have no entries below it.

The value of the **aliasedObjectName** attribute does not have to be an existing entry. However, an error will be returned when dereferencing the alias if the value of the **aliasedObjectName** attribute does not refer to an entry in the same backend as the alias entry. The value cannot be the distinguished name of the alias entry; in other words, an alias entry cannot dereference to itself.

Dereferencing an alias

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. This creates a new DN that must then be checked to see if it contains an alias that needs to be dereferenced. This process continues until the final dereferenced DN contains no alias within its name. An error will be returned if a circular chain is detected, that is, when a particular alias entry is encountered more than once. The final dereferenced DN

must be the DN of an entry in the same backend as the original DN. This entry must either exist or be under a referral entry.

Alias dereferencing is performed only during search operations. Alias entries are not dereferenced for any other LDAP operation.

Aliases are not dereferenced when performing a null-based subtree search since all entries in all LDBM, TDBM, and CDBM backends are included in the search scope.

Duplicate objects will not be returned by a search operation. Duplicate objects can be encountered during a search if an alias points to an entry higher in the tree or if two aliases point to the same entry.

Dereferencing is only used to determine the entries that will be included in the search. The entries actually returned as search results must match the search filter. The DN of returned entries is the dereferenced DN. Using the above example, a search for "cn=John Doe, ou=LDAPZOS,o=IBM" will return an entry with DN "cn=John Doe,ou=DeptC8NG,ou=Poughkeepsie,o=IBM,c=US" if the "cn=John Doe,ou=DeptC8NG,ou=Poughkeepsie,o=IBM,c=US" entry matches the search filter.

Access checking is not performed when dereferencing an alias entry. Normal access checking will be performed for the dereferenced entry. Therefore, a search can dereference aliases even though the requester 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.

Dereferencing and root DSE subtree search

Aliases are never dereferenced when performing a subtree search starting at the root DSE (this is also known as a null-based subtree search). All alias entries are processed like 'normal' entries, as if **LDAP_DEREF_NEVER** was specified.

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')

Alias examples

The following figure shows the directory structure used in the examples. The dashed lines indicate aliases. The dashed oval indicates the position of an aliased entry in the directory hierarchy, but the aliased entry does not actually exist.

Note: Fictitious attributetypes are used in the figure.

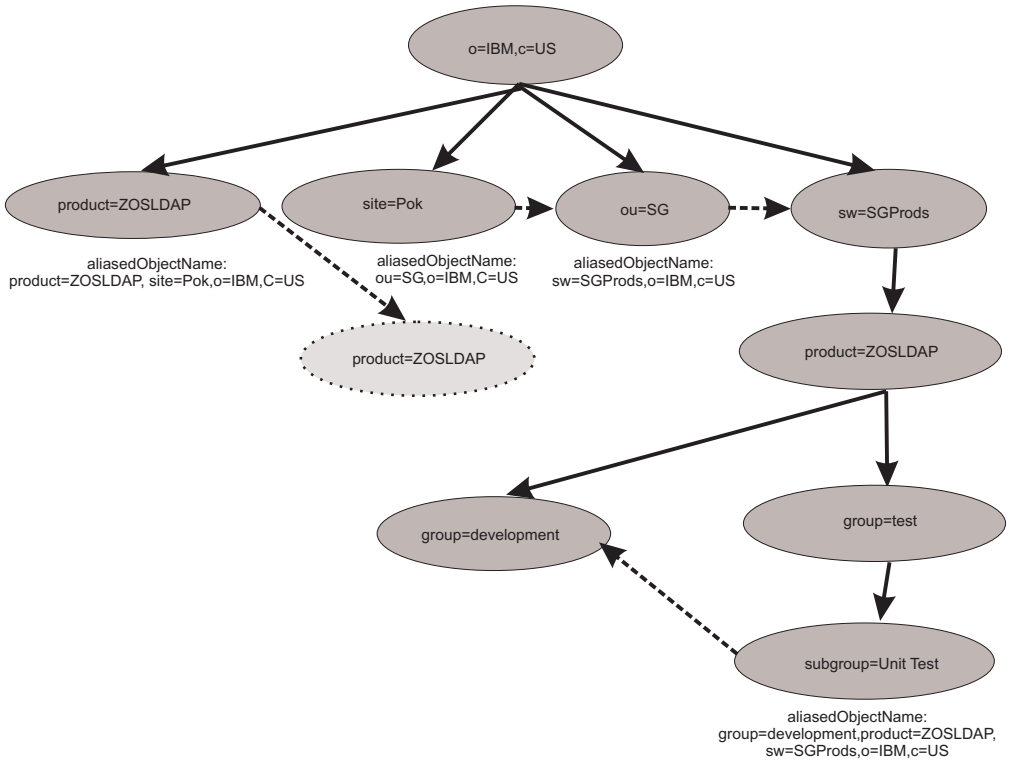


Figure 53. Alias example

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 28. 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, LDBM, or CDBM entry, to the LDAP server schema entry (`cn=schema`), or to an object controlled by an application (for example, a RACF user, group, user-group connection, or general resource 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. Therefore, 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 the GDBM 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, SDBM, or LDBM 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, SDBM, or LDBM backend, however, this is not suggested because that suffix has to be renamed to avoid an overlap if GDBM is configured in the future.

Change logging is enabled by configuring GDBM in the LDAP server configuration file. Change log processing is controlled by configuration options in the GDBM backend. The **changeLogging** configuration option turns change logging on or 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 83 for more information. If the **changeLogMaxEntries** and **changeLogMaxAge** configuration options are not specified or are both set to 0, there are no limits on the size of the change log. With this configuration, the change log must be periodically manually pruned by an LDAP root or directory data administrator to prevent it from exhausting all available space in the z/OS UNIX System Services directory or in DB2.

The **changeLoggingParticipant** configuration option can be used to specify if an LDBM, TDBM, or CDBM backend wants change log entries to be created for changes to entries in its backend. Similarly, the configuration option can be specified in the GDBM backend to determine if a change log entry should be created for a change to the LDAP server schema. If the option is not specified for a TDBM, LDBM, CDBM or GDBM backend, the default is to create change log entries for changes to that TDBM, LDBM, or CDBM backend or to the LDAP server schema.

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 a propagated ACL that allows an LDAP root or directory data administrator update access to the change log. All other administrators have read access to 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. The **aclEntry** and **entryOwner** attributes are the only attributes that can be modified. The **aclPropagate** and **ownerPropagate** attributes will always be TRUE.

Modifications to the change log are not logged. This means that no change sequence number will be returned for a persistent search request issued for the change log (`cn=changelog`).

Configuring the GDBM backend

Note: You can use the LDAP configuration utility, **dsconfig**, to configure GDBM.

The GDBM backend is configured in one of two ways: DB2-based (like TDBM) or file-based (like LDBM). In either configuration:

1. There can be at most one GDBM backend in the configuration file.
2. The **suffix** option cannot be specified in the GDBM backend.
3. If the **changeLoggingParticipant** option is specified, it controls whether a change log entry is created for a change to the LDAP server schema. Change log entries are never created for any changes to GDBM entries, including the suffix entry.

Configuring a DB2-based GDBM backend

When configuring a DB2-based GDBM backend, the following configuration file options are required:

```
database GDBM GLDBGD31/GLDBGD64 [name]
dbuserid dbowner
```

The **aclSourceCacheSize, attrOverflowSize, dnToEidCacheSize, dsnaoini, entryCacheSize, entryOwnerCacheSize, filterCacheBypassLimit, filterCacheSize, include, multiserver, persistentSearch, readonly, serverName, sizeLimit**, and **timeLimit** options can also be specified in the GDBM configuration section. The **changeLogging, changeLoggingParticipant, changeLogMaxAge**, and **changeLogMaxEntries** configuration options can be specified to control change logging activity. See Chapter 8, "Customizing the LDAP server configuration," on page 83 for more information about these options.

When using DB2 to store its entries, the GDBM database is identical to a TDBM database and is created in the same way using the same SPUFI script. A DB2-based GDBM backend cannot share a database with a TDBM backend. If the GDBM (DB2-based) backend is run in 64-bit mode, then DB2 version 9 with PTF UK50918 or DB2 version 10 or higher is required.

Configuring a file-based GDBM backend

When configuring a file-based GDBM backend, the following configuration file options are required:

```
database GDBM GLDBGD31/GLDBGD64 [name]
```

The **commitCheckpointEntries, commitCheckpointTOD, databaseDirectory, fileTerminate, filterCacheBypassLimit, filterCacheSize, include, multiserver, persistentSearch, readOnly, sizeLimit**, and **timeLimit** are options can also be specified in the GDBM configuration section. The **changeLogging, changeLoggingParticipant, changeLogMaxAge**, and **changeLogMaxEntries**

configuration options can be specified to control change logging activity. See Chapter 8, “Customizing the LDAP server configuration,” on page 83 for more information about these options.

When using files to store its entries, the GDBM database is identical to an LDBM database and is created in the same way. Like LDBM, a file-based GDBM backend can run in 64-bit mode.

Additional required configuration

Additional configuration is required for RACF to be able to log changes to a RACF user, group, connection, or resource profile:

- 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, group, connection, or resource profile. SDBM is also needed to retrieve the RACF user's new password or other changed fields. The following option must be specified in the SDBM section of the configuration file to allow change log entries to be created for changes to resource profiles:

  ```
  enableResources on
  ```

- LDAP Program Call 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
  ```

  Note: This listen parameter for LDAP Program Call support is in addition to any other listen parameters you have specified.

There is no additional configuration needed to log changes to a TDBM, LDBM, or CDBM entry or to the LDAP server schema entry. If you do not want to create change log entries for changes to entries within a TDBM, LDBM, or CDBM backend, add the following configuration option to that backend section. You can add the same option to the GDBM section of the configuration file to stop the creation of change log entries for changes to the LDAP server schema entry:

```
changeLoggingParticipant off
```

When changes are logged

Change log records can be created when the change logging is activated and the GDBM backend is not in read-only mode.

RACF changes

An extended operation, **changeLogAddEntry**, 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, group, user-group connection, or general resource profile when the profile is added, modified, or deleted. The RACF changes can be driven through the LDAP server or be made directly to RACF. For a user password or password phrase change, RACF includes information that the password or password phrase changed in the change log entry. For other user changes, RACF does not 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 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, LDBM, CDBM, and schema changes

If change logging is activated, each add, modify, delete, or modify DN operation of an entry in any TDBM, LDBM, or CDBM backend or modify of the LDAP server schema entry results in the creation of a change log entry, except for the following:

- Change log entries are not created for entries that are added to a TDBM backend when using the load utility, **ldif2ds**.
- If the **changeLoggingParticipant off** option in the LDAP server configuration file is specified for this backend, then no changes in this backend are logged. The option can be specified for the GDBM backend to stop logging changes to the LDAP server schema entry.

The change log entry is created after the change to the TDBM, LDBM, or CDBM backend entry or the LDAP server schema has been committed. This change is not rolled back if the change log record cannot be created.

Change log schema

The following object classes and their attributes define a change log entry. These object classes and attributes are always in the LDAP server schema.

- 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 user, group, class, or resource name that is passed in by RACF and the SDBM suffix.

  changeType
  add | **modify** | **delete** | **modrdn**

  changeTime
  the time stamp of when the change is made (not when this entry is created)

  changes
  the added entry or the modifications, in LDIF format. This is fully supported for change log entries created by TDBM, LDBM, CDBM, and the LDAP server schema. However, the values for the **userPassword, secretKey, replicaCredentials, ibm-slapdAdminPw, ibm-slapdMasterPw**, and **ibm-replicaKeyPwd** attributes are replaced with `*ComeAndGetIt*` in the change log entry. For change log entries created by RACF, this attribute is only present when a RACF user password or password phrase is changed, and contains either `*ComeAndGetIt*` or `*NoEnvelope*`, for example:

  ```
  replace: racfPassword
  racfPassword: *ComeAndGetIt*
  -
  ```

  newRDN
  the new RDN specified in a TDBM, LDBM, or CDBM modify DN operation

  deleteOldRdn
  a boolean indicating if the old RDN was deleted in a TDBM, LDBM, or CDBM modify DN operation

newSuperior
the new superior distinguished name specified in a TDBM, LDBM, or CDBM modify DN operation

- objectclass: **ibm-changelog**

ibm-changeInitiatorsName
the DN of the entity that initiated the change. For RACF, this DN is created from a user ID that is passed in by RACF and the SDBM suffix.

Note: If a RACF user's password or password phrase is changed using the *currentvalue/newvalue* support on a bind to the SDBM backend or on a bind using native authentication, the **ibm-changeInitiatorsName** value is created from the user ID under which the LDAP server is running (and not the bound user).

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 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 an LDAP root or directory data administrator update access to the change log. All other administrators have read access to the change log. An appropriately authorized user can modify the root entry to change the ACL. Operations on the change log root are not replicated and do not result in the creation of a change log entry.

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. Only the **aclEntry** and **entryOwner** attributes can be modified. When GDBM is configured to be file-based, the **aclEntry** and **entryOwner** attributes can be entirely deleted, in which case the default ACL is used. See “Default ACLs with LDBM or TDBM” on page 443 for more information. When GDBM is configured to be DB2-based, these attributes cannot be entirely deleted. 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 if 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.

- Change log entries are only created by the LDAP server. The user cannot directly add a change log entry. Also, the user cannot modify or rename a change log entry. Change log entries inherit the ACL of the change log root entry.
- Change log entries are deleted by the LDAP server when the change log is trimmed because of 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, compare, delete) on change log entries are allowed if 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.
- If the GDBM backend is in read-only mode, delete and modify operations are not allowed. Add and trim operations by the LDAP server are not performed.
- Operations on change log entries are not replicated and do not result in the creation of change log entries.

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: racfPassPhrase
racfPassPhrase: *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 if 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 values of the **userpassword, secretkey, replicacredentials, ibm-replicakeypwd, ibm-slapdadminpw, ibm-slapdmasterpw, racfpassword**, and **racfpassphrase** attributes are replaced by `*ComeAndGetIt*`. You can use a search command to retrieve the password. For a RACF password or password phrase, see Chapter 17, "Accessing RACF information," on page 327 for more information.

Unloading and loading the change log

The unload utility (**ds2ldif**) 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 load utility (**ldif2ds**) fail when processing change log entries.

Trimming the change log

When change logging is on, the LDAP server periodically trims the change log based on the limits set in the LDAP server 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. The number of entries that are removed depends on how GDBM is configured. When GDBM is file-based, entries are removed until the number of entries remaining is at the limit. When GDBM is DB2-based, entries are removed until the number of entries remaining 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 (for a DB2-based change log only) and when change log entries are added. When DB2-based, 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. No trimming is performed when the GDBM backend is in read-only mode. Trimming is performed when the LDAP server is in maintenance mode (and GDBM is not read-only).

Change log information in the root DSE entry

The following attributes in the root DSE 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.

Multi-server considerations

When running GDBM in multi-server mode, the change log is shared by all of the LDAP servers in the same cross-system group. Each LDAP server must have an identical GDBM backend configured to avoid the possibility of not logging a change that should be logged (or the reverse). Note that each TDBM, LDBM, and CDBM backend must run in the same mode as GDBM. They must all run in multi-server mode or all not run in multi-server mode.

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 you want. GDBM can be configured to be DB2-based or file-based. The following example starts change logging using a DB2-based change log with a maximum size of 1000 entries. Entries are automatically deleted when they become a day old.

      ```
      database gdbm GLDBGD31/GLDBGD64
      dbuserid dbu1
      changeLogging on
      changeLogMaxEntries 1000
      changeLogMaxAge 86400
      ```

   b. If you plan to log changes to RACF users, groups, user-group connections, and general resource profiles, you must also:

      Add the SDBM backend section, including the **suffix** and, optionally, the **enableResources** configuration options. The **enableResources** configuration option is only needed when logging changes to resource profiles. Following is an example:

      ```
      database sdbm GLDBSD31/GLDBSD64
      suffix cn=myRacf
      enableResources on
      ```

      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, LDBM, or CDBM backend or to the LDAP server schema entry, add the following option to the TDBM, LDBM, CDBM, or GDBM backend section (the GDBM backend controls change logging for the schema entry):

      ```
      changeLoggingParticipant off
      ```

2. If GDBM is DB2-based, create the DB2 database to be used by the change log. This involves updating and executing a SPUFI script. The database owner in the script 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 55 for more information.
3. If you plan to log changes to RACF users, groups, connections, and resource profiles, perform the RACF configuration required to support creation of an LDAP change log entry for RACF changes to those profiles. If you plan to retrieve RACF password or password phrase envelopes, you need to perform

the RACF configuration required to support creation and retrieval of the password or password phrase envelopes. See Activating LDAP change notification for more information.

4. Restart the LDAP directory server. For a DB2-based GDBM backend in 64-bit mode, this will look similar to the following.

   Note: The GDBM-0002 is a backend name assigned by the LDAP server or is the name specified on the **database** configuration option for the GDBM backend:

   ```
   database GDBM GLDBGD64 GDBM-0002
   aclSourceCacheSize: 100
   attrOverflowSize: 255
   changeLogging: on
   changeLogMaxAge: 86400
   changeLogMaxEntries: 1000
   changeLoggingParticipant: on
   dbUserid: DBU1
   dnToEidCacheSize: 1000
   entryCacheSize: 5000
   entryOwnerCacheSize: 100
   filterCacheBypassLimit: 100
   filterCacheSize: 5000
   multiserver: off
   persistentSearch: off
   readOnly: off
   sizeLimit: 1000
   suffix 1: CN=CHANGELOG
   timeLimit: 3600
   ```

   For a file-based GDBM backend in 31-bit mode, this will look similar to:

   ```
   database GDBM GLDBGD31 GDBM-0002
   changeLogging: on
   changeLogMaxAge: 86400
   changeLogMaxEntries: 1000
   changeLoggingParticipant: on
   commitCheckpointEntries: 10000
   commitCheckpointTOD: 00:00
   databaseDirectory: /var/ldap/gdbm
   fileTerminate: recover
   multiserver: off
   persistentSearch: off
   readOnly: off
   sizeLimit: 1000
   suffix 1: CN=CHANGELOG
   timeLimit: 3600
   ```

   If GDBM fails to start, the following message is issued:

   ```
   GLD1106E GDBM-0002 backend initialization failed.
   ```

5. At this point, change logging is started. Depending on your configuration, a change to a RACF user, group, connection, or resource profile, or to a TDBM, LDBM, or CDBM entry, or to the LDAP server schema entry will result in the creation of a change log entry in the LDAP server.

6. If you want, modify the ACL on the change log root entry, `cn=changelog`, for your usage of the change log. The initial ACL restricts update access to the change log to an LDAP root or directory data administrator. All other administrators have read access to the change log.

   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  -h ldaphost -p ldapport  -D adminDn  -w adminPw   -f cl.ldif
```

7. You can search, delete, and compare 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 -h ldaphost -p ldapport  -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 any of the following lines:

   ```
   racfPassword: *NoEnvelope*
   racfPassword: *ComeAndGetIt*
   racfPassPhrase: *NoEnvelope*
   racfPassPhrase: *ComeAndGetIt*
   userpassword: *ComeAndGetIt*
   replicacredentials: *ComeAndGetIt*
   secretkey: *ComeAndGetIt*
   ibm-slapdadminpw: *ComeAndGetIt*
   ibm-slapdmasterpw: *ComeAndGetIt*
   ibm-replicakeypwd: *ComeAndGetIt*
   ```

   then a password in the RACF user profile, TDBM, LDBM, or CDBM entry was changed. If the value is `*ComeAndGetIt*`, then you can try to retrieve the actual password value. See “Passwords in change log entries” on page 569 for information about retrieving passwords.

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 -h ldaphost -p ldapport  -D adminDn -w adminPw -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 29. 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 topic describes how to create referral entries in a TDBM or LDBM backend and how to configure a default referral for the LDAP server.

A referral entry can be added to a TDBM or LDBM backend to indicate that the backend does not contain that entry or any entries below it and to identify another LDAP server that may contain those entries. A referral entry returns referral information to the LDAP client if the target of a client operation is at or below the referral entry or if a search operation includes the referral entry within its search scope.

A default referral can be added to the LDAP server configuration file to identify another LDAP server that may contain entries that do not fall within any of the suffixes in this LDAP server. If the target of an operation is not at or below any suffix defined in the LDAP server, the LDAP server returns the default referral to the client.

This topic also contains information about how to associate multiple servers using referrals and an example of associating a set of servers through referrals, basic replication (see Chapter 25, "Basic replication," on page 469), and advanced replication (see Chapter 26, "Advanced replication," on page 487).

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 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.

A referral entry must be a leaf entry. This means that no ancestor of an entry can be a referral entry. In addition, a referral entry cannot also be an alias entry.

Creating referral entries

Following is an example configuration that illustrates the use of the **ref** attribute.

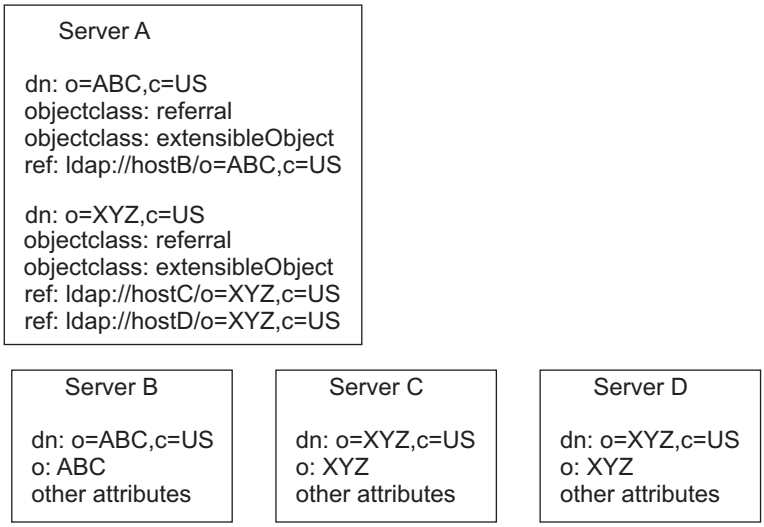


Figure 54. Example using ref attribute

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 preferred 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 entries 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 entries to point to the other servers for subordinate references that are portions of the namespace below this server which are not serviced directly.

Referral entries are created in TDBM and LDBM backends. Referral entries consist of:

dn Specifies the distinguished name. It is the portion of the namespace served by the referenced server.

objectclass
Specifies **referral**. Also, include the object class **extensibleObject**.

ref

Specifies the location of the referenced server. There is no required format for the value, however, the z/OS LDAP client can only follow a **ref** value which is in LDAP URL format. An LDAP URL has one of the following formats:

```
ldap://hostname:port/DN
ldaps://hostname:port/DN
```

The default port (389 for a non-SSL connection or 636 for an SSL connection) is used if a port is not specified as part of the LDAP URL. The DN of the referral entry is used if a DN is not specified as part of the LDAP URL. The **ldap://** form is for a non-SSL connection while the **ldaps://** form is for an SSL connection. The **ldaps://** form is required if you are using non-standard ports and want to allow SSL connections to the referenced server. The DN value in the LDAP URL should match the DN of the referral entry. 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. Note that you cannot specify a 0-length value for the **ref** attribute.

The z/OS LDAP server automatically adds the **extensibleObject** object class to a referral entry if it is not specified. This allows the RDN attributes to be added to the referral entry.

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
```

A TDBM or LDBM backend can contain any number of referral entries in its directory.

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 is specified using the **referral** option in the LDAP server configuration file and applies to all backends in the LDAP server. The value of the option must be 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.

The default referral LDAP URL does not include the DN portion and a DN, if specified, is ignored. The default port (389 for a non-SSL connection or 636 for an SSL connection) is used if a port is not specified as part of the LDAP URL. The **ldap://** form is for a non-SSL connection while the **ldaps://** form is for an SSL

connection. The **ldaps://** form is required if you are using non-standard ports and want to allow SSL connections to the referenced server. Following is an example:

```
referral  ldap://host3.ibm.com:999
```

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 LDAP clients request information from LDAP 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. In this case, the application retrieves the referral from the LDAP client and processes it within the application. 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

Referrals are not supported by the LDAP Version 2 protocol. In order to provide referral information to LDAP Version 2 clients, the referral information is returned as part of the error string in the result message. Since clients do not generally examine the error string for results indicating **LDAP_SUCCESS**, the LDAP server returns **LDAP_PARTIAL_RESULTS** instead of **LDAP_SUCCESS** if referral information is present in the error string. Referral information may be present for any result other than **LDAP_SUCCESS**.

The referral information in the error string is returned as follows, where '\n' indicates a newline character:

```
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.

Limitations with LDAP Version 2 referrals

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 entries were found in the referencing server that matched the search criteria. If chasing referrals, the client contacts every server presented in the list to continue the search request. For referral entries 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.

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" entries in the namespace, as opposed to the referral entries themselves. For searches, referral entries are only presented as referrals, since the usual intent of a search is to look at the real entries in the namespace. Server administrators must therefore use other means to examine existing referral entries, such as examining the database, or reviewing **ds2ldif** 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 entry are not followed for update operations, rather they operate on the referral entry itself. This is necessary to allow an administrator the ability to delete or modify existing referral entries. 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.

If the target of a request is a referral entry or is below a referral entry, or if the target does not fall within any of the suffixes in the LDAP server and a default referral is configured, then 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.

If the target of a search is found in the directory but a referral entry is encountered during the rest of a one-level or subtree search, a referral is not returned. 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.

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 entry or a referral entry. For LDAP Version 3, this difficulty is overcome with a protocol extension in the form of the **manageDsaIT** control. (Appendix C, "Supported server controls," on page 679 describes **manageDsaIT** in detail.) For typical client requests where the control is absent, whenever the server encounters an applicable referral entry 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 entries as normal entries. In this case, even superior

references through the use of default referrals are suppressed. The z/OS LDAP client operations utilities support the **-M** option to indicate that the requester is managing the namespace, and therefore wants to examine and manipulate referral entries as if they were normal entries. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information. An exception to the processing described above is that referral entries 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 685 for more information.

Bind considerations for referrals

When LDAP clients chase referrals from one server to another, they typically need to bind to the referenced server before redirecting the original request. If you distribute your directory across multiple servers connected by referrals, you must consider the capabilities of the applications which access your directory, how they chase referrals, and how they can bind to the referenced servers.

For example, the z/OS LDAP client utilities like **ldapsearch** and **ldapmodify** use the bind DN and password specified on the utility invocation, both when binding to the original target server and also when chasing referrals to other servers. If you want the LDAP client utilities to automatically chase referrals across servers, then the same bind DN and password must be accepted on each of the servers connected by referrals.

If you use an approach where there are no common bind identities, then your applications will either be limited to unauthenticated access or they will require the ability to bind appropriately to each server when chasing referrals.

Consider the following approaches:

1. Use unauthenticated access for reading information to avoid the need to bind with a common identity. This makes sense if the data in the directory is general reference information that does not need to be protected.
2. Establish an 'authentication' backend for identity information that is the same on each server. This could be an SDBM backend, where the common authentication identities are in RACF, or a TDBM or LDBM backend that is the same on each server (replication could be used to ensure this). Access control over the other entries in the referral servers uses the distinguished names from the authentication backend to control access to the entries.
3. If you use the LDAP root administrator DN (**adminDN** configuration option) to access the entries, configure the administrator DN and password identically in each of the referral servers.

Example: associating servers through referrals and basic 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.

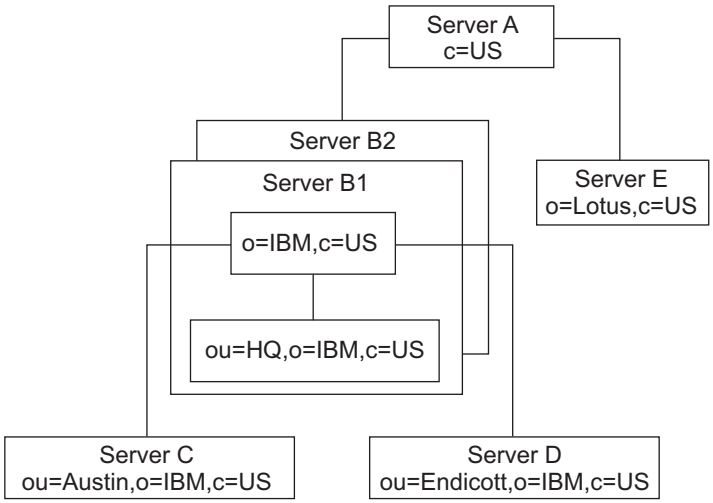


Figure 55. Setting up the servers

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 entries to point to the descendants in other servers.

```
dn: o=IBM,c=US                                  <-- Pointer to Servers B1 and B2
objectClass: referral
objectClass: extensibleObject
ref: ldap://ibm.com:389/o=IBM,c=US
ref: ldap://ibm.com:390/o=IBM,c=US

dn: o=Lotus,c=US                                <-- Pointer to Server E
objectClass: referral
objectClass: extensibleObject
ref: ldap://lotus.com:389/o=Lotus,c=US
```

Figure 56. 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 GLDBTD31/GLDBTD64
suffix                  “ou=Endicott,o=IBM,c=US”
```

```
#     General Section

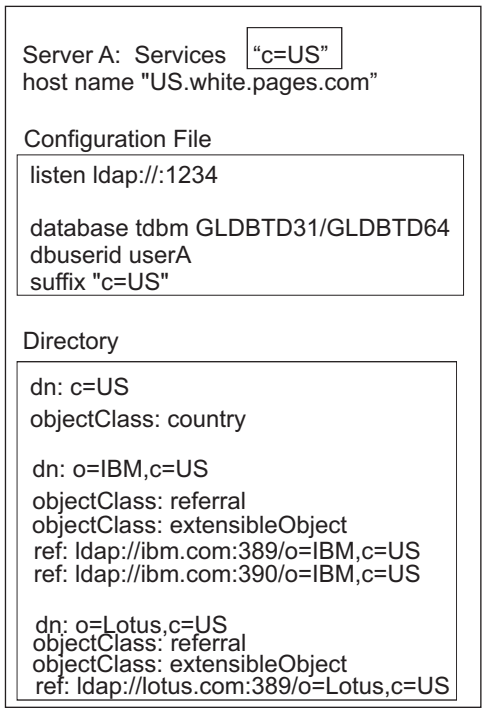
referral                ldap://ibm.com:389
referral                ldap://ibm.com:390

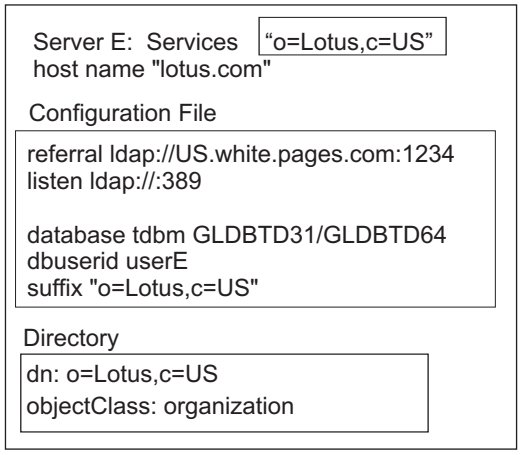
listen                  ldap://:789
.
.
.
# ldbm database definitions
database                ldbm GLDBLD31
suffix                  “ou=Endicott,o=IBM,c=US”
```

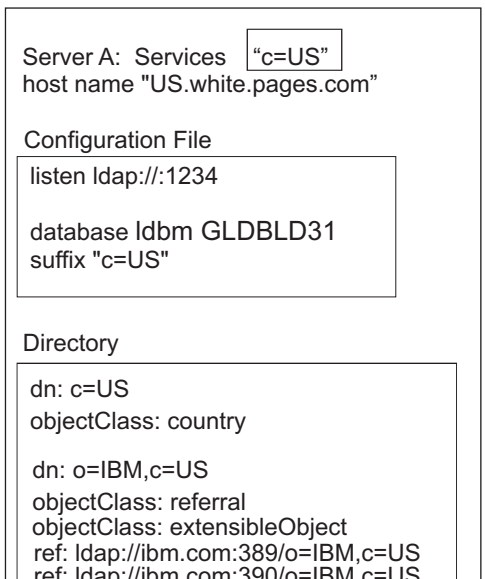
Figure 57. Server D configuration file

5. Putting it all together.

   Figure 58 on page 583, Figure 59 on page 585, and Figure 61 on page 589 show these same six servers, showing the referral entries in the database including 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 remain identical. Servers B1 and B2 are on the same system, but use different ports.


Server A: Services "c=US"
host name "US.white.pages.com"
Configuration File
listen ldap://:1234
database tdbm GLDBTD31/GLDBTD64
dbuserid userA
suffix "c=US"
Directory
dn: c=US
objectClass: country
dn: o=IBM,c=US
objectClass: referral
objectClass: extensibleObject
ref: ldap://ibm.com:389/o=IBM,c=US
ref: ldap://ibm.com:390/o=IBM,c=US
dn: o=Lotus,c=US
objectClass: referral
objectClass: extensibleObject
ref: ldap://lotus.com:389/o=Lotus,c=US


Server E: Services "o=Lotus,c=US"
host name "lotus.com"
Configuration File
referral ldap://US.white.pages.com:1234
listen ldap://:389
database tdbm GLDBTD31/GLDBTD64
dbuserid userE
suffix "o=Lotus,c=US"
Directory
dn: o=Lotus,c=US
objectClass: organization


Server A: Services "c=US"
host name "US.white.pages.com"
Configuration File
listen ldap://:1234
database ldbm GLDBLD31
suffix "c=US"
Directory
dn: c=US
objectClass: country
dn: o=IBM,c=US
objectClass: referral
objectClass: extensibleObject
ref: ldap://ibm.com:389/o=IBM,c=US

Server B1: Services "o=IBM,c=US"
host name "ibm.com"

Configuration File

```
referral   ldap://US.white.pages.com:1234
listen ldap://:389

database tdbm GLDBTD31/GLDBTD64
dbuserid userB1
suffix "o=IBM,c=US"
suffix "cn=localhost"
```

Directory

```
dn: cn=localhost
objectClass: container

dn: cn=ReplicaB2,cn=localhost
objectClass: replicaObject
replicaHost: ibm.com
replicaPort: 390
replicaBindDN: cn=Master
replicaCredentials: secret

dn: o=IBM,c=US
objectClass: organization

dn: ou=Austin,o=IBM,c=US
objectClass: referral
objectClass: extensibleObject
ref: ldap://austin.com:389/ou=Austin,o=IBM,c=US

dn: ou=Endicott,o=IBM,c=US
objectClass: referral
objectClass: extensibleObject
ref: ldap://endicott.com:789/ou=Endicott,o=IBM,c=US

dn: ou=HQ,o=IBM,c=US
objectClass: organizationalUnit
```

Server B1: Services "o=IBM,c=US"
host name "ibm.com"

Configuration File

```
referral   ldap://US.white.pages.com:1234
listen ldap://:389

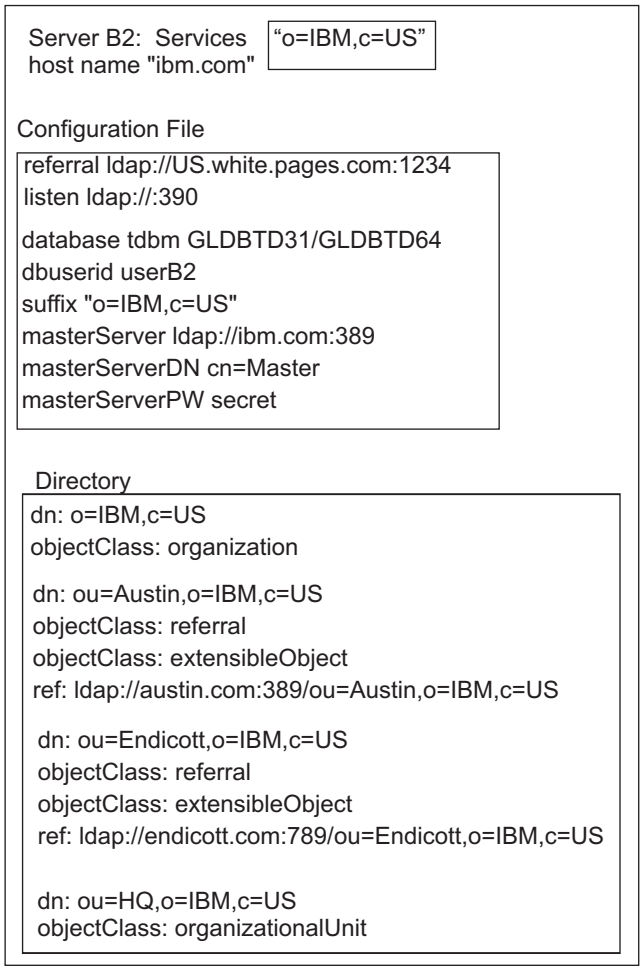
database ldbm GLDBLD31
suffix "o=IBM,c=US"
suffix "cn=localhost"
```

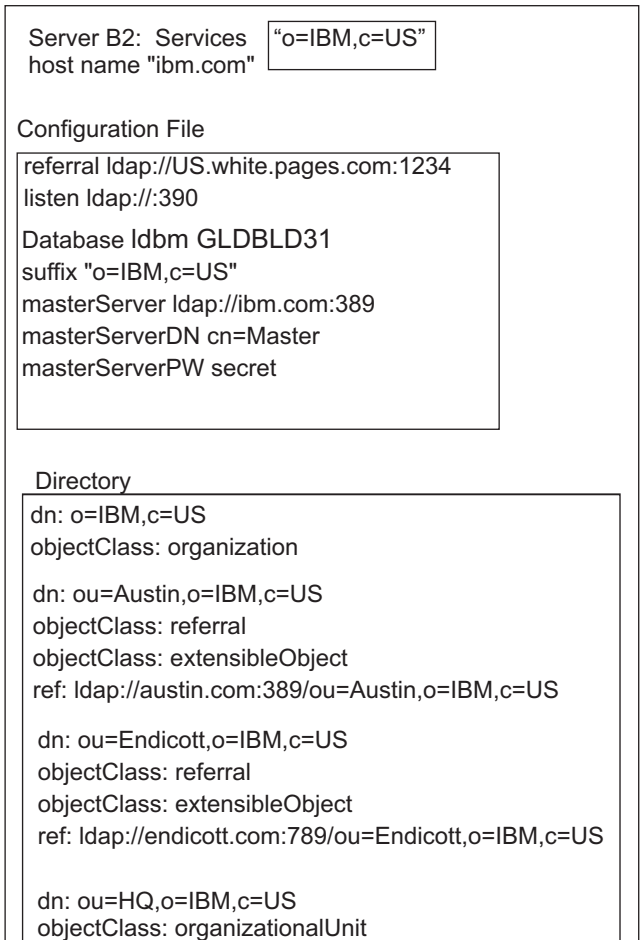
Directory

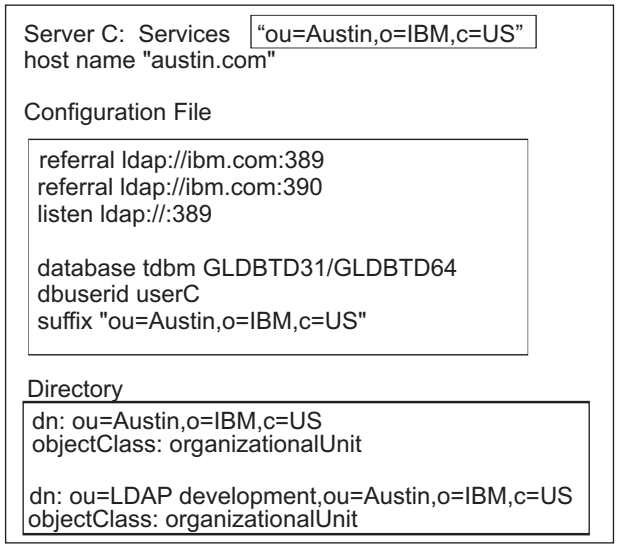
```
dn: cn=localhost
objectClass: container

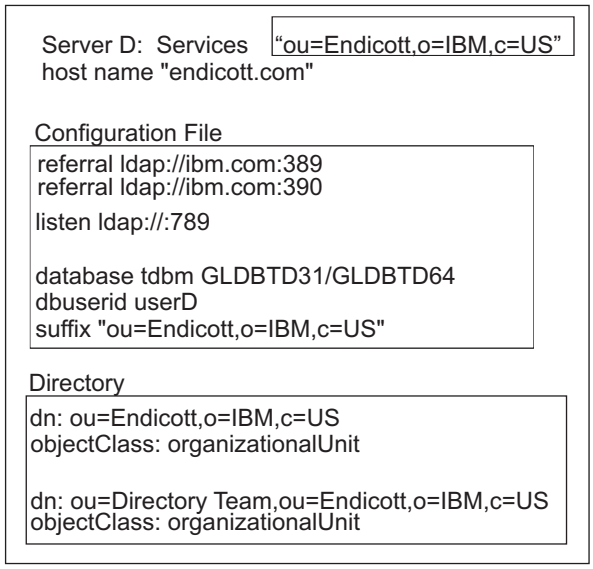
dn: cn=ReplicaB2,cn=localhost
objectClass: replicaObject
replicaHost: ibm.com
replicaPort: 390
replicaBindDN: cn=Master
replicaCredentials: secret

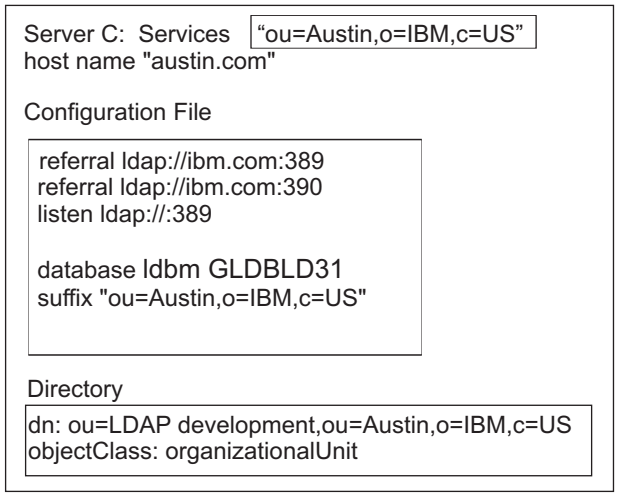
dn: o=IBM,c=US
objectClass: organization
```


Server B2: Services
host name "ibm.com"
“o=IBM,c=US”
Configuration File
referral ldap://US.white.pages.com:1234
listen ldap://:390
database tdbm GLDBTD31/GLDBTD64
dbuserid userB2
suffix "o=IBM,c=US"
masterServer ldap://ibm.com:389
masterServerDN cn=Master
masterServerPW secret
Directory
dn: o=IBM,c=US
objectClass: organization
dn: ou=Austin,o=IBM,c=US
objectClass: referral
objectClass: extensibleObject
ref: ldap://austin.com:389/ou=Austin,o=IBM,c=US
dn: ou=Endicott,o=IBM,c=US
objectClass: referral
objectClass: extensibleObject
ref: ldap://endicott.com:789/ou=Endicott,o=IBM,c=US
dn: ou=HQ,o=IBM,c=US
objectClass: organizationalUnit


Server B2: Services
host name "ibm.com"
“o=IBM,c=US”
Configuration File
referral ldap://US.white.pages.com:1234
listen ldap://:390
Database ldbm GLDBLD31
suffix "o=IBM,c=US"
masterServer ldap://ibm.com:389
masterServerDN cn=Master
masterServerPW secret
Directory
dn: o=IBM,c=US
objectClass: organization
dn: ou=Austin,o=IBM,c=US
objectClass: referral
objectClass: extensibleObject
ref: ldap://austin.com:389/ou=Austin,o=IBM,c=US
dn: ou=Endicott,o=IBM,c=US
objectClass: referral
objectClass: extensibleObject
ref: ldap://endicott.com:789/ou=Endicott,o=IBM,c=US
dn: ou=HQ,o=IBM,c=US
objectClass: organizationalUnit


Server C: Services
“ou=Austin,o=IBM,c=US”
host name "austin.com"
Configuration File
referral ldap://ibm.com:389
referral ldap://ibm.com:390
listen ldap://:389
database tdbm GLDBTD31/GLDBTD64
dbuserid userC
suffix "ou=Austin,o=IBM,c=US"
Directory
dn: ou=Austin,o=IBM,c=US
objectClass: organizationalUnit
dn: ou=LDAP development,ou=Austin,o=IBM,c=US
objectClass: organizationalUnit


Server D: Services
“ou=Endicott,o=IBM,c=US”
host name "endicott.com"
Configuration File
referral ldap://ibm.com:389
referral ldap://ibm.com:390
listen ldap://:789
database tdbm GLDBTD31/GLDBTD64
dbuserid userD
suffix "ou=Endicott,o=IBM,c=US"
Directory
dn: ou=Endicott,o=IBM,c=US
objectClass: organizationalUnit
dn: ou=Directory Team,ou=Endicott,o=IBM,c=US
objectClass: organizationalUnit


Server C: Services
“ou=Austin,o=IBM,c=US”
host name "austin.com"
Configuration File
referral ldap://ibm.com:389
referral ldap://ibm.com:390
listen ldap://:389
database ldbm GLDBLD31
suffix "ou=Austin,o=IBM,c=US"
Directory
dn: ou=LDAP development,ou=Austin,o=IBM,c=US
objectClass: organizationalUnit

Chapter 30. Client considerations

When an LDAP application is communicating with an LDAP server, consider the following special topics:

- Root DSE
- Monitor Support
- UTF-8 data over the LDAP Version 2 protocol
- Attribute types stored and retrieved in lowercase
- Abandon behavior

Root DSE

The root DSE is the entry at the top of the LDAP server directory information tree. All the **namingcontexts** (suffixes) in the LDAP server are directly below the root DSE. The root DSE contains information about the LDAP server, including the **namingcontexts** that are configured and the capabilities of the server.

The root DSE can be searched by specifying a zero-length base distinguished name. The search scope can be either base or subtree (the one-level scope is not supported).

Root DSE search with base scope

A root DSE search with base scope returns the contents of the root DSE. The root DSE attributes describe the LDAP server. The only search filter that is supported is **objectclass=***. There is no access control checking for the root DSE, but an anonymous bind fails if **allowAnonymousBinds off** is specified in the LDAP server configuration file. The **supportedcontrol**, **supportedextension**, and **namingcontexts** attributes may contain values that are contributed by plug-in extensions that are configured in the LDAP server.

The following example uses the **ldapsearch** utility to request a base search of the root DSE and shows sample output for the search:

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

Following is an example of the information that the LDAP server reports on a search of the root DSE. A subset of these values might appear in your root DSE based on the server configuration choices you made.

```
vendorname=International Business Machines (IBM)
vendorversion=z/OS V2R2
ibmdirectoryversion=z/OS V2R2
ibm-serverid=Master
ibm-slapdServerCompatibilityLevel=7
altserver=ldap://host2.ibm.com:999
subschemasubentry=cn=schema
supportedldapversion=2
supportedldapversion=3
supportedcontrol=1.3.18.0.2.10.20
supportedcontrol=2.16.840.1.113730.3.4.3
supportedcontrol=2.16.840.1.113730.3.4.2
supportedcontrol=1.3.18.0.2.10.10
supportedcontrol=1.3.18.0.2.10.11
supportedcontrol=1.3.18.0.2.10.15
supportedcontrol=1.3.18.0.2.10.18
supportedcontrol=1.3.18.0.2.10.19
supportedcontrol=1.2.840.113556.1.4.319
supportedcontrol=1.2.840.113556.1.4.473
supportedcontrol=1.3.18.0.2.10.2
supportedcontrol=1.3.6.1.4.1.42.2.27.8.5.1
```

```
supportedcontrol=1.3.18.0.2.10.23
supportedcontrol=1.3.18.0.2.10.27
supportedcontrol=1.3.18.0.2.10.24
supportedcontrol=1.3.18.0.2.10.34
supportedextension=1.3.6.1.4.1.1466.20037
supportedextension=1.3.18.0.2.12.62
supportedextension=1.3.18.0.2.12.48
supportedextension=1.3.18.0.2.12.82
supportedextension=1.3.18.0.2.12.75
supportedextension=1.3.18.0.2.12.58
supportedextension=1.3.18.0.2.12.37
supportedextension=1.3.18.0.2.12.15
supportedextension=1.3.18.0.2.12.16
supportedextension=1.3.18.0.2.12.17
supportedextension=1.3.18.0.2.12.19
supportedextension=1.3.18.0.2.12.54
supportedextension=1.3.18.0.2.12.56
namingcontexts=CN=CONFIGURATION
namingcontexts=CN=IBMPOLICIES
namingcontexts=CN=CHANGELOG
namingcontexts=CN=MYRACF
namingcontexts=O=IBM,C=US
namingcontexts=SECAUTHORITY=DEFAULT
ibm-supportedcapabilities=1.3.18.0.2.32.24
ibm-supportedcapabilities=1.3.18.0.2.32.26
ibm-supportedcapabilities=1.3.18.0.2.32.30
ibm-supportedcapabilities=1.3.18.0.2.32.28
ibm-supportedcapabilities=1.3.18.0.2.32.7
ibm-supportedcapabilities=1.3.18.0.2.32.98
ibm-supportedcapabilities=1.3.6.1.4.1.4203.1.5.1
ibm-supportedcapabilities=1.3.18.0.2.32.3
ibm-supportedcapabilities=1.3.18.0.2.32.33
ibm-supportedcapabilities=1.3.18.0.2.32.34
ibm-supportedcapabilities=1.3.18.0.2.32.31
ibm-supportedcapabilities=1.3.18.0.2.32.63
ibm-supportedcapabilities=1.3.18.0.2.32.17
ibm-supportedcapabilities=1.3.18.0.2.32.19
ibm-supportedcapabilities=1.3.18.0.2.32.5
ibm-supportedcapabilities=1.3.18.0.2.32.54
ibm-supportedcapabilities=1.3.18.0.2.32.57
ibm-supportedcapabilities=1.3.18.0.2.32.77
ibm-supportedcapabilities=1.3.18.0.2.32.88
ibm-supportedCapabilities=1.3.18.0.2.32.68
ibm-supportedcapabilities=1.3.18.0.2.32.94
ibm-supportedcapabilities=1.3.18.0.2.32.1
ibm-supportedcapabilities=1.3.18.0.2.32.29
ibm-supportedcapabilities=1.3.18.0.2.32.18
ibm-supportedcapabilities=1.3.18.0.2.32.44
ibm-supportedcapabilities=1.3.18.0.2.32.51
ibm-supportedcapabilities=1.3.18.0.2.32.52
ibm-supportedcapabilities=1.3.18.0.2.32.56
ibm-supportedcapabilities=1.3.18.0.2.32.65
ibm-supportedcapabilities=1.3.18.0.2.32.43
ibm-supportedcapabilities=1.3.18.0.2.32.2
ibm-supportedcapabilities=1.3.18.0.2.32.95
ibm-supportedcapabilities=1.3.18.0.2.32.6
ibm-supportedcapabilities=1.3.18.0.2.32.99
ibm-supportedcapabilities=1.3.18.0.2.32.105
ibm-enabledcapabilities=1.3.18.0.2.32.24
ibm-enabledcapabilities=1.3.18.0.2.32.26
ibm-enabledcapabilities=1.3.18.0.2.32.7
ibm-enabledcapabilities=1.3.6.1.4.1.4203.1.5.1
ibm-enabledcapabilities=1.3.18.0.2.32.98
ibm-enabledcapabilities=1.3.18.0.2.32.3
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=1.3.18.0.2.32.56
ibm-enabledcapabilities=1.3.18.0.2.32.2
ibm-enabledcapabilities=1.3.18.0.2.32.5
ibm-enabledcapabilities=1.3.18.0.2.32.54
ibm-enabledcapabilities=1.3.18.0.2.32.57
ibm-enabledcapabilities=1.3.18.0.2.32.77
ibm-enabledcapabilities=1.3.18.0.2.32.88
ibm-enabledCapabilities=1.3.18.0.2.32.68
ibm-enabledcapabilities=1.3.18.0.2.32.28
ibm-enabledcapabilities=1.3.18.0.2.32.17
ibm-enabledcapabilities=1.3.18.0.2.32.94
ibm-enabledcapabilities=1.3.18.0.2.32.6
ibm-enabledcapabilities=1.3.18.0.2.32.1
```

```
ibm-enabledcapabilities=1.3.18.0.2.32.29
ibm-enabledcapabilities=1.3.18.0.2.32.18
ibm-enabledcapabilities=1.3.18.0.2.32.44
ibm-enabledcapabilities=1.3.18.0.2.32.51
ibm-enabledcapabilities=1.3.18.0.2.32.52
ibm-enabledcapabilities=1.3.18.0.2.32.65
ibm-enabledcapabilities=1.3.18.0.2.32.43
ibm-enabledcapabilities=1.3.18.0.2.32.95
ibm-enabledcapabilities=1.3.18.0.2.32.105
ref=ldap://hostk.ibm.com:391
supportedsaslmechanisms=CRAM-MD5
supportedsaslmechanisms=DIGEST-MD5
supportedsaslmechanisms=EXTERNAL
ibm-sasldigestrealmname=MYHOST.IBM.COM
changelog=cn=changelog
firstchangenumber=24213
lastchangenumber=24322
```

Following are Object Identifiers (OIDs) for supported and enabled capabilities:

Table 83. Object Identifiers (OIDs) for supported and enabled capabilities

OID assigned	Short name	Description
1.3.6.1.4.1.4203.1.5.1	Retrieval of operational attributes	Indicates that this server supports the **+** attribute on search requests to return operational attributes.
1.3.18.0.2.32.1	Advanced replication	Identifies that this server supports advanced replication that includes subtree and cascading replication.
1.3.18.0.2.32.2	Entry Checksum	Indicates that this server supports the **ibm-entryCheckSum** and **ibm-entryCheckSumOp** operational attributes.
1.3.18.0.2.32.3	Entry UUID	Identifies that this server supports the **ibm-entryuuid** operational attribute.
1.3.18.0.2.32.5	Password policy	Indicates that this server supports password policies.
1.3.18.0.2.32.6	Sort by DN	Indicates that this server supports using the **ibm-slapdDN** attribute to sort by DN.
1.3.18.0.2.32.7	System restricted ACL support	Indicates that the server supports specification and evaluation of ACLs on system and restricted attributes.
1.3.18.0.2.32.17	Group search limits	Indicates that this server supports using groups to specify search size and time limits.
1.3.18.0.2.32.18	**cn=ibmpolicies** advanced replication subtree	Indicates that this server supports the replication of the **cn=ibmpolicies** subtree. This support is only available when advanced replication is configured.
1.3.18.0.2.32.19	Max age ChangeLog entries	Specifies that the server can retain changelog entries that are based on age.
1.3.18.0.2.32.24	Monitor operation counts	The server provides monitor operation counts for initiated and completed operation types.

Table 83. Object Identifiers (OIDs) for supported and enabled capabilities (continued)

OID assigned	Short name	Description
1.3.18.0.2.32.26	Null-based subtree search	Indicates that the server supports null-based subtree search operations, which search all the LDBM, TDBM, and CDBM entries in the server.
1.3.18.0.2.32.28	TLS capabilities	Specifies that the server can perform Transport Layer Security (TLS).
1.3.18.0.2.32.29	Non-blocking advanced replication	Indicates that this server can ignore some errors that are received from a consumer (replica) server that would normally cause an update to be retransmitted periodically until a successful return code is received.
1.3.18.0.2.32.30	Kerberos capability	Specifies that the server can perform Kerberos authentication.
1.3.18.0.2.32.31	**ibm-allMembers** and **ibm-allGroups** operational attributes	Indicates that a backend supports searching on the **ibm-allGroups** and **ibm-allMembers** operational attributes. The members of a static, dynamic, or group can be obtained by performing a search on the **ibm-allMembers** operational attribute. The static, dynamic, and nested groups that a member DN belongs to can be obtained by performing a search on the **ibm-allGroups** operational attribute.
1.3.18.0.2.32.33	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.
1.3.18.0.2.32.34	Modify DN (subtree rename)	Indicates that a subtree can be renamed. The DN of each entry under the subtree is also changed. This rename uses a new RDN but not a new superior.
1.3.18.0.2.32.43	Advanced replication configuration	Indicates that this server supports configuration of supplier servers in an advanced replication environment.
1.3.18.0.2.32.44	Global updates support	Indicates that this server supports the advanced replication of global updates using the replication topology in the **cn=ibmpolicies** subtree in the CDBM backend.

Table 83. Object Identifiers (OIDs) for supported and enabled capabilities (continued)

OID assigned	Short name	Description
1.3.18.0.2.32.51	Advanced replication conflict resolution maximum entry size	Indicates that this server supports the **ibm-slapdReplConflictMaxEntrySize** attribute on a CDBM entry with an objectclass of **ibm-slapdReplicationConfiguration**. This attribute value indicates the maximum number of bytes that an entry can contain and still can be resent to a target server as a result of advanced replication conflict resolution.
1.3.18.0.2.32.52	Lost and found log	Indicates that this server supports the lost and found log for archiving replaced entries as a result of the advanced replication conflict resolution.
1.3.18.0.2.32.54	Password policy account lockout	Indicates that this server supports the password policy account lockout feature.
1.3.18.0.2.32.56	Updated **ibm-entryCheckSumOp** operational attribute calculation	Indicates that this server supports an updated algorithm for the checksum calculation of the **ibm-entryCheckSumOp** operational attribute.
1.3.18.0.2.32.57	LDAP password global start time	Indicates that the server supports the **ibm-pwdPolicyStartTime** attribute in the **cn=pwdpolicy,cn=ibmpolicies** entry.
1.3.18.0.2.32.63	Salted SHA (SSHA)	Indicates that this server supports the Salted SHA hashing of password values.
1.3.18.0.2.32.65	Filter replication	Identifies that this server supports filtered replication that allows only required entries and a subset of attributes to be replicated. This support is only available when advanced replication is configured.
1.3.18.0.2.32.68	Administrative roles	Indicates that this server allows administrative roles to be defined and used for administrative group members.
1.3.18.0.2.32.77	Multiple password policies	Indicates that this server allows multiple password policy entries to be defined and used.
1.3.18.0.2.32.88	Password policy maximum consecutive repeated characters	Indicates that this server supports password policies that restrict the maximum number of consecutive repeated characters in password values.

Table 83. Object Identifiers (OIDs) for supported and enabled capabilities (continued)

OID assigned	Short name	Description
1.3.18.0.2.32.94	Fine grained time stamps	Indicates that this server supports advanced replication with fine grained time stamps that include microseconds.
1.3.18.0.2.32.95	**ibm-replicationWaitOnDependency** attribute replication	Indicates that this server supports the replication of the **ibm-replicationWaitOnDependency** attribute from the advanced replication agreement entry.
1.3.18.0.2.32.98	ACL filter support	Indicates that this server supports specifying a filter in the access control attributes to further control access to an object.
1.3.18.0.2.32.99	SHA-2 and Salted SHA-2 hashing	Indicates that this server supports SHA-2 and Salted SHA-2 hashing.
1.3.18.0.2.32.105	Replication of security attributes	Indicates that the server supports replication of security attributes between master and read-only replica so that password policy for account lockout is enforced in replication topologies.

Root DSE search with subtree scope (Null-based subtree search)

A root DSE search with subtree scope returns all the entries that match the search filter in the LDBM, TDBM, and CDBM backends that are configured in the LDAP server. This search is commonly referred to as a null-based subtree search. Note that the search does not include the root DSE itself, the LDAP server schema entry, SDBM entries, and GDBM entries (change log records). Alias entries are not dereferenced during the search, they are processed like normal entries and returned if they match the search filter. Referral entries in LDBM, TDBM, and CDBM return referrals to the client. Any filter can be specified for the subtree search. A sorted root DSE search with subtree scope sorts the entire result set after all entries have been retrieved from the backends.

A null-based subtree is implemented as a series of searches to each LDBM, TDBM, and CDBM suffix. These individual searches are each limited by the time limit and size limit options that are specified in the LDAP server configuration file or in the requester's group search limits. If a time limit or size limit is specified on the root DSE search request, then the individual searches are also limited by the amount of time remaining and the number of entries that are left to return when that individual search is started. See the descriptions of the **sizeLimit** and **timeLimit** options in Chapter 8, “Customizing the LDAP server configuration,” on page 83 for more information. See “Managing group search limits” on page 423 for more information about group search limits. Each individual LDBM, TDBM, and CDBM search is subject to the normal LDBM, TDBM, and CDBM access control checking.

The following example uses the **ldapsearch** utility to request a subtree search of the root DSE for entries that have a `cn` value that begins with `ken` and shows sample output for the search.

```
ldapsearch -h ldaphost -p ldapport -D binddn -w passwd -s sub -b "" "cn=ken*"

cn=ken,o=ldbm
objectclass=person
objectclass=top
cn=ken
sn=smith

cn=kenx,o=tdbm
objectclass=person
objectclass=top
cn=kenx
sn=jones
```

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" on page 612.

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 LDAP 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. See 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

The LDAP server reads additional operations as they arrive if 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.

Chapter 31. Performance tuning

Overview

Several server configuration options and facilities significantly affect the performance of the server. In addition, specific LDAP server backends operate with other products that might require tuning to accommodate the LDAP server. For example, the TDBM and DB2-based GDBM backends use DB2, that provides a large set of tuning options. The SDBM backend provides access to the RACF database, that has its own product-specific tuning options. This topic 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 that 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.

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 DB2-based GDBM is configured), and other system resources associated with threads. Therefore, you might want to avoid making this option larger than is needed.

Set the **commThreads** to approximately two times the number of processors 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 directory 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 affects performance. If optimal performance is what you want, active debug only when necessary to capture diagnostic information.

Storage in the LDAP address space

The LDAP server generally requires a minimum of 32 MB to run in 31-bit mode, and 96 MB in 64-bit mode even with a minimal directory. This storage is required for maintaining server-wide information and for processing client requests.

Note: These are estimates only, and the need for storage can increase depending on the size of any LDBM directories configured, and the size of the caches.

LDAP server cache tuning

The LDAP server implements many caches to help reduce processing time and to avoid access to the 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 might increase the amount of storage that is required by the server.

Some caches are invalidated by update activities. If this is a frequent occurrence, increasing the cache size might 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, therefore, you should generally avoid disabling them unless close monitoring is done to ensure that they are not beneficial.

Most caches in the LDAP server are enabled by default, and the default sizes 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. These are described later in this topic.

  Note: The monitor search must be used with a scope of subtree or one-level to retrieve the cache statistics because 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. Because 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 the LDAP server:

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. This cache is implemented in the TDBM and DB2-based GDBM backends.

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. This is a server-wide cache, and is implemented in the internal schema backend. To alter its setting from the default, adjust the **dnCacheSize** configuration option in the global section of the LDAP server configuration file.

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. This cache is implemented in the TDBM and DB2-based GDBM backends.

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. This cache is implemented in the TDBM and DB2-based GDBM backends.

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. This cache is implemented in the TDBM and DB2-based GDBM backends.

filter cache
This cache holds information related to the mapping of search request inputs and the result set. This cache is implemented in the TDBM, LDBM, CDBM, and GDBM backends. For TDBM and DB2-based GDBM, retrieval of information from this cache avoids database read operations when processing search requests. For LDBM, CDBM, and file-based GDBM, this cache helps reduce processing time for searches with complex filtering. Note that the GDBM filter cache is disabled, by default.

Operations monitor

If the operations monitor is enabled, the LDAP server monitors search statistics for the types of search patterns that are configured and stores search statistics for each search pattern. The operations monitor supports two types of search patterns, **searchStats** and **searchIPStats**. A **searchStats** pattern consists of the search parameters (search base, scope, filter, and attributes to be returned) and status (success or failure). The **searchStats** pattern is useful for evaluating the performance of search patterns. A **searchIPStats** pattern consists of the same elements as **searchStats** pattern does, but also includes the client IP address. The **searchIPStats** pattern is useful in determining if there are any specific clients spamming the LDAP server. The **operationsMonitor** configuration option determines which types of search patterns are monitored. See "Monitoring performance with cn=monitor" on page 612 for more information about the operations monitor.

A new search pattern is added to the operations monitor whenever the search pattern of an incoming search does not match one of the existing operations monitor search patterns. When the number of search patterns exceeds the value of the **operationsMonitorSize** configuration option (the **cachesize** attribute in the **cn=operations,cn=monitor** entry), the least recently used search patterns are trimmed. The total number of trimmed search patterns is stored in the **numtrimmed** attribute of the **cn=operations,cn=monitor** entry. Typically, trimmed search patterns are not a cause for concern because they are infrequently executed search patterns. If there is a high volume of trimmed data, you should consider increasing the value of the **operationsMonitorSize** configuration option or possibly monitoring only **searchStats** patterns. Note that **searchIPStats** search patterns produce more search patterns than **searchStats** patterns because **searchIPStats** creates a new search pattern for each unique client IP address even if the rest of the search pattern is the same. See Table 88 on page 617 for more information about the **cn=operations,cn=monitor** entry and its attributes.

Workload manager (WLM)

The z/OS LDAP server supports Workload Manager (WLM) to allow an installation to set performance goals for work within the LDAP server when compared against other work that is running on the system. The **LDAP** subsystem type is reserved in WLM to allow the system administrator to set performance goals for z/OS LDAP server operations. The performance goals can be set based on the IP address of the client, distinguished name (DN) of the bound user, both the IP address and DN associated with the request, or a request matching a search pattern in the operations monitor.

The z/OS LDAP WLM support is always active and cannot be deactivated. A default service class must be configured in the WLM ISPF panels for the **LDAP** subsystem type before running the z/OS LDAP server. If a default service class is not configured in WLM, all LDAP server operations run under the discretionary or SYSOTHER profile and receive a low priority, which impacts LDAP server performance and response times. See *z/OS MVS Planning: Workload Management* for more information about WLM. See "Verifying the service class for the LDAP server" on page 604 for information about determining the service class for the LDAP servers that are running on your system.

The LDAP server uses the WLM health service to indicate a health value to WLM. The WLM health value is used by the TCP/IP sysplex distributor to help route incoming client requests to servers within the sysplex. After the LDAP server initialization, the WLM health value is set to 100%. The WLM health value is calculated by the number of failures during the past 5000 operations if one minute has passed since the value was last calculated. If the percentage of failures changes by 25% or more, the z/OS LDAP server increases or decreases the WLM health value. An LDAP server operation is considered a failure when it has one of the following return codes:

LDAP_OPERATIONS_ERROR (1)
LDAP_TIMELIMIT_EXCEEDED (3)
LDAP_ADMIN_LIMIT_EXCEEDED (11)
LDAP_BUSY (51)
LDAP_UNAVAILABLE (52)
LDAP_UNWILLING_TO_PERFORM (53)
LDAP_OTHER (80)

The **wlmExcept** configuration option can be used to specify the IP address of the client, the distinguished name (DN) of the bound user, or both to route those requests to any configured WLM transaction name (TN). The **wlmExcept** configuration option can be specified multiple times within the LDAP server configuration file to allow multiple client IP addresses or bound users' DNs to be associated with the same or different WLM transaction name. The order that the **wlmExcept** configuration options are specified in the LDAP server configuration file determines the order the LDAP server uses to match incoming client requests and route them to the WLM transaction name. See "wlmExcept name [IP_address] [dn]" on page 137 for more information about the **wlmExcept** configuration option.

The **WLMEXCEPT** operator modify command can be used to change the routing of incoming client requests to new or different WLM transaction names while the server is running. If the operations monitor is configured, the **cn=operations,cn=monitor** entry has **searchStats** and **searchIPStats** attribute values with an ID parameter that indicates the operations monitor ID (OPID). See "Monitoring performance with cn=monitor" on page 612 for details about the **searchStats** and **searchIPStats** attribute value format. The **WLMEXCEPT** operator

modify command uses the OPID to associate a search pattern to a WLM transaction name. If the transaction name specified on the **WLMEXCEPT** operator modify command does not exist in WLM, a new WLM enclave is created; otherwise an existing enclave is used. Each time the **WLMEXCEPT** operator modify command is issued, the new mappings are added before any of the configured **wlmExcept** configuration options or previously issued **WLMEXCEPT** operator modify commands. See "LDAP server operator commands" on page 205 for more information about the **WLMEXCEPT** operator modify command.

The **WLMEXCEPT** operator modify commands last for the life of the LDAP server, however, the **RESET WLMEXCEPT** can be issued to remove all previously issued **WLMEXCEPT** operator modify commands and default to using the initial LDAP server configuration. If the operations monitor ID (OPID) is specified on the **RESET WLMEXCEPT** operator modify command, then just that specific WLM routing for that search is removed. See "LDAP server operator commands" on page 205 for more information about the **RESET WLMEXCEPT** operator modify command.

If the operations monitor is enabled, the **searchStats** and **searchIPStats** attributes in the **cn=operations,cn=monitor** entry can be used to identify spamming client applications or certain search requests that should have a higher priority within the LDAP server. This type of information is very valuable when configuring LDAP to use WLM transaction names and assigning service or report classes to those transaction names. For a spamming client application, a WLM transaction name with a low priority service or report class ought to be used. For important search requests, a WLM transaction name with a high priority service or report class ought to be used.

Configuring LDAP with WLM examples

Assume that WLM has been configured to contain the following transaction names and service classes for LDAP.

```
Subsystem-Type  Xref  Notes  Options  Help
------------------------------------------------------------------------------
                Modify Rules for the Subsystem Type       Row 1 to 3 of 3
Command ===> _____________________________________________ Scroll ===> PAGE

 Subsystem Type . : LDAP        Fold qualifier names?   Y  (Y or N)
 Description  . . . Use Modify to enter YOUR rules

 Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                 B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                          More ===>
          --------Qualifier--------               -------Class--------
 Action    Type       Name     Start                Service     Report
                                              DEFAULTS: HIGHREQ     ________
   ____  1  TN         GENERAL  ___                  NORMREQ     ________
   ____  1  TN         EXCEPT1  ___                  SPAMREQ     ________
   ____  1  TN         EXCEPT2  ___                  CRITREQ     ________
****************************** BOTTOM OF DATA ******************************


  F1=Help    F2=Split   F3=Exit    F4=Return  F7=Up      F8=Down    F9=Swap
 F10=Left   F11=Right  F12=Cancel
```

Figure 62. WLM ISPF rules classification panel

Example 1:

After analyzing the **searchStats** and **searchIPStats** attributes returned on a **cn=operations,cn=monitor** search, it has been determined there is a spamming LDAP client application on IP address `1.2.3.4` that has been affecting performance

of the LDAP server. Also, requests from bound user cn=importantguy,o=ibm should have a higher priority within the LDAP server.

An LDAP administrator can add the following **wlmExcept** configuration options to route these requests to the appropriate WLM transaction name in Figure 62 on page 603:

```
wlmExcept EXCEPT1 1.2.3.4
wlmExcept EXCEPT2 cn=importantguy,o=ibm
```

After the LDAP server is restarted, LDAP client requests originating from IP address 1.2.3.4 are routed to WLM transaction name EXCEPT1 with a service class of SPAMREQ. The server routes requests from bound user cn=importantguy,o=ibm to WLM transaction name EXCEPT2 with a service class of CRITREQ. If there are requests from any other client IP addresses or bound users, the server routes them to the GENERAL WLM transaction name that has a service class of NORMREQ. See *z/OS MVS Planning: Workload Management* for more information about configuring WLM.

If the transaction name specified on the **wlmExcept** configuration option or on the **WLMEXCEPT** operator modify command does not exist in WLM (such as EXCEPT3 does not exist in Figure 62 on page 603), any client requests associated with that transaction name would use the default service class HIGHREQ.

Example 2:

After analyzing the **searchStats** and **searchIPStats** attribute values returned on a **cn=operations,cn=monitor** search, it has been determined that the search identified by a specific **searchIPStats** value should be mapped to WLM transaction name EXCEPT2 because it is an important LDAP search.

```
cn: cn=monitor,cn=operations
searchIPStats: ldap://fe00::f4f7:0:0:7442:750f/OU=_v,O=_v??sub?(|(&(sn=_v)(cn=_v*))
 (description=*_v*))?success,numOps=42,avg=246,rate=5,maxRate=37,maxRat
 eTimeStamp=20130524132626.545031Z,createTimeStamp=20130524132615.953823Z,
 ID=2741
```

The following **WLMEXCEPT** operator modify command is used to route the search pattern in the **searchIPStats** attribute to the EXCEPT2 WLM transaction name:

```
F LDAPSRV,WLMEXCEPT EXCEPT2,2741
```

Note: The operations monitor ID (OPID) value is 2741 for the search on the **searchIPStats** attribute value.

If it is determined later that the client search requests identified by OPID 2741 no longer need to be routed to WLM transaction name EXCEPT2, the following **RESET WLMEXCEPT** operator modify command is used to remove just the specific WLM routing for that search:

```
F LDAPSRV,RESET WLMEXCEPT,2741
```

See “LDAP server operator commands” on page 205 for more information about the **RESET WLMEXCEPT** operator modify command.

Verifying the service class for the LDAP server

If you are having performance problems with the LDAP server, verify that the LDAP server is running with the correct service class in WLM. If a default service class is not configured in WLM, all LDAP server operations that are run under the discretionary or SYSOTHER profile and receive a low priority, which impacts LDAP server performance and response times.

Use the ISPF SDSF menu and specify ENC (enclaves) to verify the LDAP server service class. Figure 63 depicts the LDAP server is running under the SYSOTHER service class, which results in LDAP operations receiving a low priority.

```
SDSF ENCLAVE DISPLAY  DCEIMGVD ALL                    LINE 1-6 (6)
 COMMAND INPUT ===>                                        SCROLL ===> PAGE
 NP   NAME            SSType Status   SrvClass Per PGN RptClass ResGroup   CPU-
      3400000008      LDAP   ACTIVE   SYSOTHER   1
      2400000002      STC    INACTIVE SYSSTC     1
      2800000003      STC    INACTIVE SYSSTC     1
      3000000005      STC    INACTIVE SYSSTC     1
      2000000001      STC    INACTIVE SYSTEM     1
      2C00000004      TCP    INACTIVE SYSOTHER   1
```

Figure 63. SDSF enclave display example

Password policy considerations

If password policy is enabled, additional cost is incurred during authentication to enforce the policy and to update operational attributes within the user entry. Generally, authentications do not incur much additional overhead with password policy enabled unless updates to the user entry occur.

Failed authentications require updates to the user entry to record the failed attempt, and to conditionally lock the account. Most successful authentications do not cause updates. However, the first expiration warning for a given user entry requires an update to record the event. Also, the first successful authentication which follows a failed authentication for the given user entry requires an update to clear operational attributes related to preceding failed attempts.

The operational attributes that might be updated during authentication are:

- **pwdAccountLockedTime**
- **pwdExpirationWarned**
- **pwdFailureTime**
- **pwdGraceUseTime**
- **ibm-pwdAccountLocked**

See “Password policy operational attributes” on page 379 for more details about these attributes.

Use of multiple policies might cause additional cost during authentication and during password change by the user because of group membership determination. In both cases, the effective password policy must be evaluated to apply password policy rules.

To evaluate the effective password policy of a user, the LDAP server considers all the password policies associated with a user. This means that the LDAP server evaluates the individual, group, and global password policies to determine a user’s effective password policy. If you have defined individual and group password policies, the group membership of the user must be resolved to properly apply group password policies.

LDBM performance considerations

The LDAP server LDBM backend uses the z/OS UNIX System Services file system for its persistent storage of the directory entry data. When the LDAP server is executing, the entire directory contents are held in its address space, including index structures for quick access.

This provides extremely fast access to the directory data. LDAP operations that read directory data involve no DASD I/O during the operation. LDAP operations that update the directory generally perform DASD I/O only to write the changed information to the LDBM checkpoint file. The index updates only occur within the LDAP server address space, and are not stored on DASD. Compared to TDBM, LDBM operations generally run much faster and use much less processor resources.

However, LDBM has inherent scalability limitations. The following resources are affected by the size of the directory, and are generally proportional to the LDBM directory size:

- The storage required within the LDAP server address space
- The LDAP server initialization time, both elapsed time and processor time
- The time required to commit the directory
- The DASD space required for the directory, including space for commit processing.

Storage in the LDAP address space for LDBM data

Since the entire LDBM directory is kept in storage in the LDAP address space, you must plan accordingly. The amount of storage required can be estimated from the size of the LDIF data that is used to load the directory. For 31-bit mode, the storage that is needed to contain the data is about 7 to 10 times the size of the LDIF file. If you are running in 64-bit mode, storage requirements increase to as much as 10 to 15 times the size of the LDIF file.

These are estimates that pertain only to the extra storage required to hold the LDBM directory representation.

Note: When running in multi-server mode, each of the LDAP servers sharing the LDBM directory in the sysplex (known as sysplex replicas for this backend) also keeps the entire LDBM directory in its address space. Therefore, each sysplex replica requires about the same amount of storage for the LDBM directory as the LDAP server that owns the data in the sysplex (known as the sysplex master for this backend).

For large LDBM directories, it might be necessary to run in 64-bit mode. Also, you might want to consider optimizing storage use for the z/OS LDAP server. See "Tuning LE heap and heap pools" on page 626 for more information.

For systems that are constrained on storage, or for very large directories, you might need to use TDBM instead of LDBM.

LDAP server initialization time with LDBM

Whenever the LDAP server is restarted, it reads the entire LDBM directory into storage and builds the necessary index structures for efficient search processing. This can take several minutes depending on the speed of the processor, the speed of the DASD that holds the data, and the competition for resources because of

other workloads. Generally, the initialization elapsed time and the consumed processor time during initialization are proportional to the size of the directory.

When running in multi-server mode, each of the sysplex replicas receives a copy of the directory from the sysplex master using sysplex services. The amount of processor time consumed by a sysplex replica during initialization is about the same as that used by the sysplex master when it reads the directory into storage. However, the elapsed time required to initialize the sysplex replica tends to be longer than that required to initialize the sysplex master server. Initialization of a sysplex replica also consumes processor time on the sysplex master as it sends the data to the sysplex replica. The processor time consumed on the sysplex master is about one-third the processor time consumed on the sysplex replica during sysplex replica initialization.

Database commit processing

The LDBM directory contents are kept on DASD in the database files and the checkpoint file. There is one checkpoint file for the backend, and a separate database file for each suffix defined in the backend. The database files contain the overall contents of each entry in the database at the last database commit point. The checkpoint file contains individual entry updates that occurred since the last database commit point, recorded as sequential changes beyond the contents of the database file.

To avoid unbounded growth of the checkpoint file, the database is periodically committed. Commit processing writes new copies of the database and checkpoint files such that the new database files contain the up-to-date contents of each entry in the directory, and the checkpoint file contains no individual file update information. Database commits occur at the following times:

- When the number of checkpoint entries exceeds the value of the **commitCheckpointEntries** option in the LDAP server configuration file.
- When the time of day reaches the **commitCheckpointTOD** option in the LDAP server configuration option.
- When the LDAP server COMMIT operator command is invoked.
- When the LDAP server is shut down normally.
- When the LDAP server is restarted and uncommitted updates exist in the checkpoint file after an abnormal termination of the LDAP server.

Commit processing requires both processor and DASD resources, and the resources needed increase as the size of the directory increases. For large directories, commit processing might take a minute or more depending on competition for resources.

When commit processing occurs, a new copy of each directory file is created in its entirety before deleting the old copy and before deleting the previous checkpoint file. Therefore, you should plan enough DASD space to accommodate two copies of each directory file plus the maximum size of your checkpoint file. The amount of DASD space needed for the checkpoint file is highly dependent on the nature of the updates performed, and is best determined by experimentation.

During commit processing, no update requests are processed. Therefore, you should consider avoiding unplanned commits caused by the configuration option **commitCheckpointEntries**. Instead, consider using **commitCheckpointTOD**, automated methods of using the LDAP server COMMIT operator command, or planned shutdowns of the LDAP server to control when commit processing occurs.

DASD space for LDBM data

The amount of space needed to store an LDBM backend in a z/OS UNIX System Services file system is approximately four to six times the size of the expected input LDIF data. Generally, the space required to hold the LDBM backend data is two to three times the size of the expected input LDIF data. However, during the LDBM commit process each of the LDBM database files is copied, therefore, resulting in occasionally needing twice the amount of file system space.

Sample LDBM benchmark data

The following data was gathered from benchmarks using an LDBM database on z/OS UNIX System Services using ESS800 model 2105 DASD, and running on a z9® model 2094 processor with no competing applications running.

Table 84. LDBM benchmark data

LDBM benchmark data	
Database size	500,000 entries
LDIF file size	590 megabytes
LDAP server storage	5464 megabytes
Sysplex master initialization elapsed time	121 seconds
Sysplex master initialization processor time	98 seconds
Sysplex replica initialization elapsed time	180 seconds
Sysplex replica initialization processor time	103 seconds
Sysplex master processor time consumed by replica initialization	34 seconds
LDAP server database commit elapsed time	50 seconds
LDAP server database commit processor time	43 seconds

CDBM performance considerations

If the CDBM backend is only used to store configuration entries, then no tuning is necessary. However, if the CDBM backend is used to store user-defined entries, see "LDBM performance considerations" on page 606 for CDBM tuning 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 section:

- DB2 tuning to improve database access
- TDBM tuning that affects DB2 usage

DB2 tuning is important to ensure that TDBM requests that 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 that 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. See "LDAP server cache tuning" on page 600 for more information about TDBM caches.

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 by using the DB2 **REORG** utility
- Periodically maintaining the database statistics by using 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 by using the DB2 **REORG** utility. This helps to improve database access performance and to reclaim fragmented space.

DB2 Managing Performance (http://publibfp.dhe.ibm.com/epubs/pdf/dsnpgm0e.pdf) contains the DB2 real-time statistics stored procedure (DSNACCOX) that is a sample stored procedure that makes recommendations to help you maintain your DB2 databases. DSNACCOX uses data from catalog tables, including real-time statistics tables, to make its recommendations.

The need to reorganize the data is based on the amount of update activity that occurs. If a date/time based REORG strategy is chosen rather than using the real-time statistics capability of DB2, then weekly REORGs can be scheduled, or some other regularly scheduled maintenance interval can be chosen that best fits the customer environment and typical volume of updates. REORGs should be scheduled frequently enough to maintain good organization of the data as determined by the recommendations. More frequent REORGs might be beneficial if large percentage changes in inserts/updates/deletes occur. For example, when preparing for new workloads and populating the database with large amounts of data, or when introducing new types of LDAP data with new entry attributes and objectclasses, it might be beneficial to schedule a REORG after the new data is populated.

You should collect and store statistics with the **REORG** utility. Either run the **RUNSTATS** utility immediately after the **REORG** utility, or include the STATISTICS option directly on the **REORG TABLESPACE** and **REORG INDEX** utility control statements when reorganizing the data. Collecting and storing current, accurate statistics in the DB2 catalog for the TDBM database, table spaces, tables, indexes, and partitions allow DB2 to select efficient access paths to the TDBM database. The parameters you might want to specify when using the **RUNSTATS** utility are shown below:

```
//SYSIN  DD *
     RUNSTATS TABLESPACE GLDDB.SEARCHTS REPORT YES
       TABLE (LDAPTB1.DIR_SEARCH)
       INDEX (GLDSRV.DIR_SEARCHX1 KEYCARD
              FREQVAL NUMCOLS 1 COUNT 100
              FREQVAL NUMCOLS 2 COUNT 100,
             GLDSRV.DIR_SEARCHX2 KEYCARD
              FREQVAL NUMCOLS 1 COUNT 100
              FREQVAL NUMCOLS 2 COUNT 100
       COLGROUP (VALUE) FREQVAL COUNT 100
       SHRLEVEL CHANGE UPDATE ALL SORTDEVT SYSDA
     RUNSTATS TABLESPACE GLDDB.ENTRYTS REPORT YES
       TABLE (ALL)
       INDEX (GLDSRV.DIR_ENTRYX1 KEYCARD
              FREQVAL NUMCOLS 1 COUNT 100)
     RUNSTATS TABLESPACE GLDDB.DESCTS REPORT YES
       TABLE ALL INDEX ALL KEYCARD
     RUNSTATS TABLESPACE GLDDB.LENTRYTS REPORT YES
       TABLE ALL INDEX ALL KEYCARD
     RUNSTATS TABLESPACE GLDDB.LATTRTS REPORT YES
       TABLE ALL INDEX ALL KEYCARD
     RUNSTATS TABLESPACE GLDDB.MISCTS REPORT YES
       TABLE ALL INDEX ALL KEYCARD
     RUNSTATS TABLESPACE GLDDB.REPTS REPORT YES
       TABLE ALL INDEX ALL KEYCARD
  /*
```

Note:

1. In the example above, `GLDDB` is the TDBM database name and `GLDSRV` is the user ID used to create the TDBM tables and indexes. `GLDSRV` is the same value that is used in the LDAP server configuration file for **dbuserid**.
2. In the example above:
   - `GLDSRV.DIR_SEARCHX1` is the index on columns `ATTR_ID,VALUE,EID` of table `GLDSRV.DIR_SEARCH`.
   - `GLDSRV.DIR_SEARCHX2` is the index on columns `EID,ATTR_ID` of table `GLDSRV.DIR_SEARCH`.
   - `GLDSRV.DIR_ENTRYX1` is the index on columns `PEID,EID` of table `GLDSRV.DIR_ENTRY`.
3. In the example above, the values that are specified for `COUNT` are intended as guidelines and might need to be updated depending on the data in the LDAP TDBM database. These numbers represent a minimum frequency. If there is a large amount of data in a TDBM database and potentially many frequent values, these numbers might need to be increased until the optimum frequency is found.
4. When the LDAP server is started, it examines the statistics that are recorded by the **RUNSTATS** utility in the DB2 catalog. Informational messages are issued to the server output file detailing the column statistics. In addition, messages are issued if any statistics appear to be insufficient. See LDAP database definitions in your SPUFI files to correlate the table space and index names in the **RUNSTATS** input with the table and column names that appear in these messages. If you run the DB2 **RUNSTATS** utility after the LDAP server has completed initialization, you can use the LDAP server **REFRESH DB2RUNSTATS** operator modify command to reexamine the statistics.

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 suggested that the **REORG** and **RUNSTATS** utilities be run immediately after the directory is populated with this initial data

before rollout to production. If the initial population of data is done by using an application (as opposed to the **ldif2ds** 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 might be needed to ensure that DB2 uses efficient access paths based on the statistics that are gathered from the database, once it contains a representative amount of information. Without this information, poor access paths might be chosen that 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 that 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 might help ensure that the index buffers remain in the buffer pools. This technique might 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 for z/OS are especially useful for monitoring buffer pool activity.

See "Partitioning DB2 tables for TDBM" on page 57 for more information about increasing TDBM performance for DB2 partitioning if you have a large directory and do many update operations on the directory.

TDBM database tuning

Several choices might ultimately affect the performance of TDBM when accessing its data within DB2:

- The LOCKSIZE chosen on the TDBM table spaces can be important if you perform many database updates. The default LOCKSIZE of ANY is generally preferred, and is typically sufficient if you perform mostly query activity and low volumes of updates to the database. This usually results in PAGE locking, that causes locking of rows for directory entries other than the one being updated. However, if you have high volumes of update activity, you might experience DB2 deadlocks in the TDBM database with PAGE locking. If this occurs, you might want to set LOCKSIZE ROW on the TDBM table spaces that contains the **DIR_ENTRY** and the **DIR_SEARCH** table.
- 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 by way of 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 suggested 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 correct length during initial setup of the directory. Changing the size requires the **DIR_ENTRY** table to be redefined, and 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 by 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 correct length during initial setup of the directory. Changing the size requires the **DIR_SEARCH** table to be redefined, and the directory must be unloaded and reloaded to implement the change.

- The **attrOverflowSize** value specified in the TDBM section of the LDAP server configuration file.

  This configuration option specifies the threshold size of attribute values that 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 that 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 that 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 that making this option value too small can affect search performance.

Monitoring performance with cn=monitor

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 in multiple entries:

- Server-wide statistics are contained in an entry whose distinguished name is **cn=monitor**.
- Each configured backend has statistics contained in its own entry named **cn=backend*XXXX*,cn=monitor**, where *XXXX* is the backend name specified on the **database** configuration option in the server configuration file. If no backend name is specified on the **database** configuration option, the LDAP server generates a name. The naming contexts pertaining to the specific backend are also included in the entry to identify which server backend is being reported.
- Several entries contain statistics for backends that are created by the LDAP server:
  - **cn=backendMonitor,cn=monitor** - Statistics for the backend handling **cn=monitor** searches
  - **cn=backendSchema,cn=monitor** - Statistics for the backend managing the schema
  - **cn=backendRootDSE,cn=monitor** - Statistics for the backend handling root DSE searches
- If the operations monitor is on (the **operationsMonitorSize** configuration option is not set to zero), the **cn=operations,cn=monitor** entry contains statistics on search patterns.

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 and the operations monitor entry is returned (if configured).

sub (subtree search)
All entries are returned, including the operations monitor entry (if configured).

The statistics reported on the **cn=monitor** subtree search can also be displayed by using the LDAP server DISPLAY operator modify command. Operations monitor statistics cannot be displayed by using the DISPLAY operator command. The command is:

```
f dssrv,display monitor
```

See the description of the DISPLAY MONITOR output in "Displaying performance information and server settings" on page 184 for details.

Statistics generally reflect data that is gathered since the LDAP server was started. However, many of the counters can be reset by using the LDAP server RESET operator modify command. The command is:

```
f dssrv,reset monitor
```

In this case, the values reflect data that is gathered since the last reset.

The monitor search returns the following attributes:

Table 85. Server statistics

currentconnections	Current number of client connections
currenttime	Current date and time on the server
livethreads	Configured number of communication threads (**commThreads**)
maxconnections	Configured maximum number of concurrent client connections (**maxConnections**)
maxreachedconnections	High water mark for concurrent client connections
pagedentriescached	Number of entries currently cached for paged search requests
pagedsearches	Number of paged search requests completed. This value is incremented only after the last page of a paged search request is returned
resets	Number of times statistics were reset
resettime	Date and time statistics were last reset
searchpagessent	Number of pages sent for paged search requests. This value is incremented after each page of a paged search request is returned
sortedsearches	Number of sorted search requests completed. If a paged search request was specified along with a sorted search request, this value is incremented only after the last page of the paged search request is returned
starttime	Date and time the server was started
sysmaxconnections	System defined maximum number of connections
timedoutconnections	Number of idle client connections closed based on the **idleConnectionTimeout** setting

Table 85. Server statistics (continued)

timedoutpagesets	Number of paged search result sets abandoned based on the **idleConnectionTimeout** setting
totalconnections	Number of client connections that have been made to the server
version	Version of the LDAP server

The statistics reported for the **maxconnections, sysmaxconnections, totalconnections, currentconnections**, and **maxreachedconnections** attribute values only contain information for network connections. PC connection statistics are not included in these attribute values.

The **sysmaxconnections** value might 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 reflects the system maximum connection limit. For information about how the maximum number of client connections is set in the LDAP server, see the **maxConnections** configuration option at "maxConnections num-connections " on page 113.

When statistics are reset, **resettime** is set to the value of **currenttime, resets** is incremented, **maxreachedconnections** is set to the value of **currentconnections**, and **timedoutconnections, timedoutpagesets, pagedsearches, searchpagessent**, and **sortedsearches** are set to 0. None of the other server statistics listed above are affected by a reset.

Table 86. Server and backend specific statistics

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
bytessent	Number of bytes of data sent
comparesrequested	Number of compare operations requested
comparescompleted	Number of compare operations completed
deletesrequested	Number of delete operations requested
deletescomplketed	Number of delete operations completed
entriessent	Number of search entries sent
extopsrequested	Number of extended operations requested
extopscompleted	Number of extended operations completed
groupgatheringevents	Number of group gathering events 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
opscompleted	Number of operations completed
opsinitiated	Number of operations initiated
searchreferencessent	Number of search references sent
searchesrequested	Number of search operations requested. If a paged search request was specified, this value is incremented only when the initial paged search request is received

Table 86. Server and backend specific statistics (continued)

searchescompleted	Number of search operations completed. If a paged search request was specified, this value is incremented only after the last page of a paged search request has been returned
unbindsrequested	Number of unbind operations requested
unbindscompleted	Number of unbind operations completed
unknownopsrequested	Number of unrecognized operations completed
unknownopscompleted	Number of unrecognized operations completed

When statistics are reset, all of the server and backend specific statistics listed above are set to zero.

Table 87. 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**)

Table 87. Backend specific statistics (continued)

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
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**)
namingcontexts	Suffixes managed by this backend

Note that not all cache statistics shown above appears for each backend. A backend reports statistics for those caches that it supports. The schema backend reports **dn_cache** statistics. The LDBM and CDBM backends report **filter_cache** statistics. A TDBM backend reports statistics for all caches except the **dn_cache**. A DB2-based GDBM backend reports statistics for all caches except the **dn_cache**, while a file-based GDBM backend only reports **filter_cache** statistics.

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 because of 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 until a RESET MONITOR command is issued or the server ends.

Table 88. Operations monitor statistics

cachesize	Configured maximum number of search patterns in the operations monitor (**operationsMonitorSize**)
currenttimestamp	Current date and time in Coordinated Universal Time stamp format
entries	Total number of search patterns in the operations monitor entry
numtrimmed	Number of search patterns trimmed from the operations monitor
resets	Number of times the operations monitor statistics were reset
resettimestamp	Date and time in Coordinated Universal Time stamp format of last reset or server startup if the reset command was never issued
searchStats	Search statistics for search patterns based on the search parameters (search base, scope, filter, and attributes to be returned) and status (success or failure)
searchIPStats	Search statistics for search patterns consisting of the same elements as the **searchStats** pattern, but also including the client IP address

When statistics are reset, **resetTimestamp** is set to **currentTimestamp**, **resets** is incremented by one, **entries** is set to zero, **numtrimmed** is set to zero, and all search patterns are deleted.

The Coordinated Universal Time stamp format used in the **currenttimestamp** and **resettimestamp** attribute values is:

```
yyyymmddhhiiss.uuuuuuZ
```

Where,

yyyy is year, *mm* is month, *dd* is day, *hh* is hour, *ii* is minutes, *ss* is seconds, *uuuuuu* is microseconds, Z is a character constant meaning that this time is based on Coordinated Universal Time, also known as GMT.

The **searchIPStats** and **searchStats** attribute values contain search rates and other search activity that are being monitored. Depending upon the LDAP server configuration, there can be **searchIPStats** and **searchStats** attribute values returned in the **cn=operations,cn=monitor** entry for each search executed against the LDAP server. The **searchStats** attribute values contain the total of all data collected for all searches matching this search pattern no matter the IP address of the client.

The format of the **searchIPStats** and **searchStats** attribute values is:

```
ldap://clientIP/baseDN?attributes?scope?filter-string?status,numOps=numOps,avg=avg,
rate=rate,maxRate=maxRate,maxRateTimeStamp=maxRateTimeStamp,
createTimeStamp=createTimeStamp,ID=opid
```

The following describes the LDAP search pattern parts:

attributes
: List of attributes to be returned.

avg Average elapsed time for each occurrence of search pattern in microseconds.

baseDN
Distinguished name of the base of the search, with `_v` substituted for attribute values.

clientIP
Client IP address (omitted for **searchStats** search patterns).

createTimeStamp
Date and time this search pattern was first added, in Coordinated Universal Time stamp format.

filter-string
Search filter with substitutions for literal attribute values. Excluding the `*` character, all strings in values are substituted with `_v`. For example: (`cn=*bob*bah*`) would be (`cn=*_v*_v*`). There is no substitution on **objectclass** equality values when the **objectclass** is defined in the schema.

maxRate
The highest rate on this entry.

maxRateTimeStamp
Date and time **maxRate** was last set, in Coordinated Universal Time stamp format.

numOps
Total number of times this search pattern has occurred.

opid A unique integer value that distinguishes each operations monitor search pattern.

rate Number of search operations processed in the previous one minute interval. Starting with server startup or the last reset command, rate is recalculated for each search pattern every 60 seconds.

scope **base** for base object searches, **one** for one-level searches, and **sub** for subtree searches.

status `success` for any search operation that results in return code `LDAP_SUCCESS`, `LDAP_PARTIAL_RESULTS`, or `LDAP_REFERRAL`. Any other return codes result in status being set to `failure`.

See Table 88 on page 617 for the time stamp format.

In addition to the above syntax, the following character escaping is performed:
comma = %2C
percent = %25
question mark = %3F
space = %20

Note: The comma, percent, and question mark characters are not escaped when they are used as metacharacters in the search pattern.

For information about monitoring performance with the LDAP server DISPLAY MONITOR operator command, see "Displaying performance information and server settings" on page 184.

Note: DISPLAY MONITOR output does not display **cn=operations,cn=monitor** data.

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 2 Release 2 IBM Tivoli Directory Server LDAP Server
livethreads=10
maxconnections=24982
sysmaxconnections=25000
totalconnections=20709
currentconnections=1
maxreachedconnections=15
timedoutconnections=12
timedoutpagesets=0
pagedentriescached=0
pagedsearches=5
sortedsearches=2
opsinitiated=62126
opscompleted=62125
abandonsrequested=0
abandonscompleted=0
addsrequested=2318
addscompleted=2318
bindsrequested=20709
bindscompleted=20709
comparesrequested=0
comparescompleted=0
deletesrequested=2228
deletescompleted=2228
extopsrequested=0
extopscompleted=0
modifiesrequested=11501
modifiescompleted=11501
modifydnsrequested=440
modifydnscompleted=440
searchesrequested=4222
searchescompleted=4221
unbindsrequested=20708
unbindscompleted=20708
unknownopsrequested=0
unknownopscompleted=0
groupgatheringevents=20641
entriessent=4221
bytessent=1564656734
searchreferencessent=0
searchpagessent=18
currenttime=Thu May 23 16:33:00.187846 2013
starttime=Thu May 23 15:52:21.693392 2013
resettime=Thu May 23 15:52:21.693392 2013
resets=0
```

Following is an example of output of a monitor search with scope=one for a server configured with TDBM and LDBM backends. This returns backend specific statistics and operations monitor statistics. The cache statistics shown would only be included for TDBM, LDBM, GDBM, CDBM, and schema backends, since the other backend types do not implement caches. Operations monitor statistics are included for all backends.

Note that not all operational statistics for each backend are shown in the example below. They have been omitted from the example only, and appear in full for a **cn=monitor** search.

```
ldapsearch -L -h ldaphost -p ldapport -b cn=monitor -s one objectclass=*
```

```
dn: cn=backendMyTDBM,cn=monitor
namingcontexts: C=CA
namingcontexts: C=TDBM
namingcontexts: CN=MOVER
namingcontexts: CN=MOVING
opsinitiated: 3013
opscompleted: 3013
abandonsrequested: 0
abandonscompleted: 0
```

```
addsrequested: 380
addscompleted: 380
bindsrequested: 0
bindscompleted: 0
comparesrequested: 0
comparescompleted: 0
deletesrequested: 365
deletescompleted: 365
extopsrequested: 0
extopscompleted: 0
modifiesrequested: 1645
modifiescompleted: 1645
modifydnsrequested: 63
modifydnscompleted: 63
searchesrequested: 560
searchescompleted: 560
unbindsrequested: 0
unbindscompleted: 0
unknownopsrequested: 0
groupgatheringevents: 3580
unknownopscompleted: 0
entriessent: 560
bytessent: 105692
searchreferencessent: 0
acl_source_cache_size: 100
acl_source_cache_current: 1
acl_source_cache_hit: 3012
acl_source_cache_miss: 1
acl_source_cache_percent_hit: 99.97%
acl_source_cache_refresh: 0
acl_source_cache_refresh_avgsize: 0
dn_to_eid_cache_size: 1000
dn_to_eid_cache_current: 555
dn_to_eid_cache_hit: 195263
dn_to_eid_cache_miss: 4035
dn_to_eid_cache_percent_hit: 97.98%
dn_to_eid_cache_refresh: 0
dn_to_eid_cache_refresh_avgsize: 0
entry_cache_size: 5000
entry_cache_current: 562
entry_cache_hit: 381420
entry_cache_miss: 1259
entry_cache_percent_hit: 99.67%
entry_cache_refresh: 0
entry_cache_refresh_avgsize: 0
entry_owner_cache_size: 100
entry_owner_cache_current: 1
entry_owner_cache_hit: 3012
entry_owner_cache_miss: 1
entry_owner_cache_percent_hit: 99.97%
entry_owner_cache_refresh: 0
entry_owner_cache_refresh_avgsize: 0
filter_cache_size: 5000
filter_cache_current: 0
filter_cache_hit: 0
filter_cache_miss: 0
filter_cache_percent_hit: 0.00%
filter_cache_refresh: 2446
filter_cache_refresh_avgsize: 0
filter_cache_bypass_limit: 100

dn: cn=backendLDBM-002,cn=monitor
namingcontexts: C=AU
namingcontexts: C=LDBM
...
searchreferencessent: 0
filter_cache_size: 5000
filter_cache_current: 0
filter_cache_hit: 0
filter_cache_miss: 0
filter_cache_percent_hit: 0.00%
filter_cache_refresh: 16487
filter_cache_refresh_avgsize: 0
filter_cache_bypass_limit: 100

dn: cn=backendMonitor,cn=monitor
namingcontexts: CN=MONITOR
...

dn: cn=backendSchema,cn=monitor
```

```
namingcontexts: CN=SCHEMA
...
searchreferencessent: 0
dn_cache_size: 1000
dn_cache_current: 1000
dn_cache_hit: 123743
dn_cache_miss: 22017
dn_cache_percent_hit: 84.90%
dn_cache_refresh: 0
dn_cache_refresh_avgsize: 0


dn: cn=backendRootDSE,cn=monitor
...

dn: cn=operations,cn=monitor
searchStats: ldap:///OU=_v,O=_v,C=_v?telephoneNumber,postalAddress,mail,uid?o
 ne?(objectclass=inetOrgPerson)?failure,numOps=51,avg=230,rate=32,maxRate=32,
 maxRateTimeStamp=20130524132741.415477Z,createTimeStamp=20130524132628.36161
 8Z,ID=2737
searchStats: ldap:///OU=_v,O=_v??sub?(|(&(sn=_v)(cn=_v*))(description=*_v*))?
 success,numOps=42,avg=246,rate=5,maxRate=37,maxRateTimeStamp=20130524132626.
 545031Z,createTimeStamp=20130524132615.953823Z,ID=2738
searchStats: ldap:///RACFGROUPID=_v+RACFUSERID=_v,PROFILETYPE=_v,CN=_v?racfco
 nnectowner,racfconnectgroupauthority,racfconnectgroupuacc?base?(objectClass=
 *)?success,numOps=4,avg=240,rate=0,maxRate=4,maxRateTimeStamp=20130524132628
 .047031Z,createTimeStamp=20130524132626.878552Z,ID=2739
searchIPStats: ldap://9.12.47.208/OU=_v,O=_v,C=_v?telephoneNumber,postalAddre
 ss,mail,uid?one?(objectclass=inetOrgPerson)?failure,numOps=51,avg=230,rate=3
 2,maxRate=32,maxRateTimeStamp=20130524132741.415477Z,createTimeStamp=20130524
 132628.361618Z,ID=2740
searchIPStats: ldap://fe00::f4f7:0:0:7442:750f/OU=_v,O=_v??sub?(|(&(sn=_v)(cn
 =_v*))(description=*_v*))?success,numOps=42,avg=246,rate=5,maxRate=37,maxRat
 eTimeStamp=20130524132626.545031Z,createTimeStamp=20130524132615.953823Z,ID=
 2741
searchIPStats: ldap://127.0.0.1/RACFGROUPID=_v+RACFUSERID=_v,PROFILETYPE=_v,C
 N=_v?racfconnectowner,racfconnectgroupauthority,racfconnectgroupuacc?base?(o
 bjectClass=*)?success,numOps=4,avg=240,rate=0,maxRate=4,maxRateTimeStamp=2013
 0524132628.047031Z,createTimeStamp=20130524132626.878552Z,ID=2742
currenttimestamp: 20130524132836.785259Z
resettimestamp: 20130524132615.369362Z
resets: 0
numtrimmed: 0
entries: 6
cachesize: 1000
```

User groups considerations in large directories

The LDAP server supports group definitions, allowing a collection of Distinguished Names (DNs) of user entries to be associated with the group for either access control or for application-specific uses. Three types of group definitions are supported: static, dynamic, and nested groups. Static groups define each member of the group explicitly in the group entry by populating the **member** attribute (or **uniqueMember** attribute) with the user entry DNs, one DN per attribute value. Dynamic groups define one or more search patterns using the multi-valued **memberURL** attribute, such that any entry matching the **memberURL** is a member of the group. Nested groups define group relationships and hierarchies, such that a parent group contains all the members of its nested child groups. In addition, a group can be defined as a combination of these types. See Chapter 23, “Static, dynamic, and nested groups,” on page 419 for more information.

As directories grow to include large user populations, the number and size of these user groups tend to grow as well. As a result of these increases, certain types of operations that are related to querying group membership, collecting the groups that are associated with a user, or managing the membership of groups might grow in cost depending on the type of group that is used, and the type of operation.

General performance guidelines:

- The cost of determining a user's groups increases roughly in proportion to the number of groups containing the user. This applies regardless of whether the groups are static, dynamic, or nested.
- Defining many static groups throughout the directory unrelated to the user does not generally affect the cost of determining the groups for a single user.
- Defining too many dynamic groups in the directory can adversely affect the cost of determining a user's groups, even if the user is only in a few of these groups. More precisely, this depends on the number of dynamic group **memberURL** filters that cannot be indexed. If most of the **memberURLs** can be indexed, the existence of many dynamic groups and **memberURLs** in the directory remains efficient. See "Dynamic groups memberURL filter indexing considerations" on page 623 for more information about indexing of dynamic groups.
- Complex filters in **memberURLs** require more processing than simple URLs when determining a user's group membership. Try to keep memberURL filter expressions simple:
  - Filters that apply to an attribute that distinguishes the members of the group are an appropriate use of a dynamic filter, such that the filter expression is simple and maintenance free. For example:

    ```
    (&(jobRole=sales)(division=423))  ... appropriate.
    ```

  - Enumerating individual members of a group in a filter is not an appropriate use of a dynamic group. Static groups are more appropriate when you want to enumerate individual group members. For example:

    ```
    (|(userID=000123)(userID=000147)...(userID=(094863)) ... not appropriate.
    ```

- Try to be consistent and efficient in your **memberURL** filter definitions. For example, if you choose to create equivalent filters, make them identical. Avoid differences such as:

  ```
  (&(division=123)(objectclass=person))  in one memberURL
  (&(objectclass=person)(divison=123)) in a second memberURL
  (&(objectclass=person)(divison=123)(objectclass=person))  in a third memberURL.
  ```

  The first and second differ in order. The third filter has a redundant filter part that is unnecessary compared to the second filter.
- Extremely large static groups can be difficult to administer, and costly for performance. Typical activities like gathering a user's groups, deleting a member from a large static group, or adding a member to a large static group are efficient. Administrative tasks such as deleting a large static group from the directory, adding it to the directory, or retrieving all its members can require substantial processing cost and storage, and the cost increases roughly in proportion to the size of the group. See "Large static groups considerations."
- Avoid redundant definitions. For example, if you have many static groups with mostly the same user list and a few exceptions, consider placing the common set of users in a single child nested group to reduce the overall list of users in each parent group, with only the exceptions that are enumerated in the parent group.

Large static groups considerations

Enterprises with large static groups in z/OS LDAP might experience performance problems and increased storage usage in the LDAP server as the static groups grow in size. Directories that hold large numbers of users and groups for products such as Tivoli Access Manager (TAM) and WebSphere users are susceptible to this, but any product that manages user groups in the LDAP directory by using large static groups might experience the symptoms that are outlined below.

Enterprises using these products sometimes create static groups in LDAP containing many members with every user in the registry defined in one large static group. The performance impacts might worsen as the registry grows, with any of the following symptoms:

- Increased response time in the application
- 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)

Some scenarios that require substantial amounts of processing and storage within the z/OS LDAP server, are:

- A search operation that returns all the members of a large static group. This includes either a search that returns the many values with the **member** or **uniqueMember** attribute, or a search that returns the many values in the **ibm-allMembers** operational attribute.
- A search operation that requests all the members of a large static group, but the members are not returned because ACL read permissions prevent the requester from seeing the data.
- Update requests that touch a large static group entry when **persistentSearch on** is configured for a TDBM or LDBM backend that contains the large entry.

These scenarios are also susceptible to the effects of LE HEAPPOOL usage as described below.

The addressability limits of the z/OS LDAP server might become a factor when there are hundreds of thousands or millions of members in a single static group.

In this case, consider the following corrective actions:

- If you run the LDAP server in 31 bit addressability mode, increase the region size, if possible.
- Run the LDAP server in 64 bit addressability mode, and set the MEMLIMIT parameter sufficiently large.
- Limit the number of members that are placed within a single static group and partition the users into separate static groups. The number of members for each static group that can be managed successfully depends on many factors, such as the size of the member values, the amount of virtual storage that is defined for the z/OS LDAP server, and the level of concurrent activity within the server.
- If possible, avoid configuring **persistentSearch on** for a TDBM or LDBM backend that contains large entries. Some applications that use persistent search must use the changelog, and only need **persistentSearch on** configured for the GDBM backend.

Dynamic groups memberURL filter indexing considerations

The LDAP server constructs a partial index for dynamic group **memberURL** filters to improve efficiency to determine a user's groups. All URLs that are ineligible for filter indexing must be evaluated against the user entry even if the user entry is only in a few groups. The following details pertain to the capability of indexing filters:

- Simple filters that use an equality match are eligible for indexing. For example:

  ```
  (department=123)
  ```

- Present filters are eligible for indexing. For example:

(accountID=*)

- Filters that use substring match are not eligible for indexing. For example:
  (department=HQ*)
- Filters that use greater or equal match are not eligible for indexing. For example:
  (department>=800)
- Filters that use less than or equal match are not eligible for indexing. For example:
  (department<=199)
- "AND" filters are eligible for indexing if at least one subfilter is eligible. For example:
  (&(jobTitle=sales)(department>=800)) ...

  Is eligible because (jobTitle=sales) is eligible.
  (&(department=HQ*)(department>=800)) ...

  Is not eligible because both filter parts are not eligible.
- "OR" filters are eligible for indexing only if all subfilters are eligible.
  (|(department=HQManagement)(department=HQAccounting)) ...

  Is eligible as both subfilters are eligible.
- "NOT" filters are not eligible for indexing.
  (!(department=123))

The eligibility of filters for the index does not mean that the filter is indexed. The indexing is done based on primitive subfilters within the more complex filters of type "AND" or "OR". The goal of the indexing is to avoid the need to evaluate many URLs. Therefore, if a subfilter part within the index is referenced too often, it is excluded from the indexing. For example, consider the following use of filters in the entire set of dynamic group URLs in the directory:

```
(&(department=000001)(objectclass=person))  ... memberURL filter in group for all users in department 000001
(&(department=000002)(objectclass=person))  ... memberURL filter in group for all users in department 000002
... etc.
(&(department=100000)(objectclass=person)) ... memberURL filter in group for all users in department 100000
(objectclass=person)     ... memberURL filter in the "everyone" group
(&(objectclass=person)(division>=900))  ... memberURL filter for high numbered division (>=900) employees
```

Roughly half of the potentially indexable filter parts are (objectclass=person). This is not a low frequency usage. If it is used from the index, it does not help disqualify dynamic group **memberURLs** from the need to be evaluated, as they all include this filter part. Because of its high frequency, it is not retained in the index. However, each of the 100,000 department matches is retained in the index. Therefore, all but the last two **memberURLs** are indexed. The threshold for retaining a filter part in the index is about one-eighth of the total number of unique, matchable **memberURLs**.

Consider the effects of your dynamic group **memberURL** filter definitions. If you need to define more than a few hundred dynamic groups and the filters are not eligible for indexing, consider alternatives if the performance cost becomes too great. Options include:

1. Use an alternative filter expression that is indexable.

   Consider the following filter, which is not indexable because it uses a substring filter to match multiple departments:

   (department=98*)

If just a few departments match, the following alternative might be preferable, enumerating the departments. This is indexable because each of the equality filters within the OR filter is indexable:

```
(|(department=98J)(department=98S)(department=98T))
```

2. Use special attributes within the user entries to specifically enumerate and match the dynamic groups. IBM provides the **ibm-dynamicMember** objectclass and the **ibm-group** attribute for this purpose. Both are found in the schema.user.ldif file. Steps are shown below to use this method.

   In a dynamic group referred to as `group1`, use the following filter within its **memberURL** attribute:

   ```
   (ibm-group=group1)
   ```

   The actual group entry might look like this:

   ```
   dn: cn=group1,ou=groups,o=ABCcompany
   objectclass: groupOfUrls
   memberurl: ldap:///o=ABCcompany??sub?(ibm-group=group1)
   ```

   Then, for any user you want to include in this group, add the **ibm-group** attribute with value `group1`. You also need to add the objectclass **ibm-dynamicMember** to the user entry to allow the **ibm-group** attribute. The user entry might look like:

   ```
   dn: cn=user1,ou=users,o=ABCcompany
   objectclass: person
   objectclass: ibm-dynamicMember
   ibm-group: group1
   ... other user attributes
   ```

   If you define your own user objectclass and attributes within the schema, this approach can be simplified to avoid the need for an extra objectclass in the user entries. For example, if you include attribute **ibm-group** as a **MAY** attribute in objectclass **ABCperson**, the **ibm-dynamicMember** objectclass is not needed in the user entry:

   ```
   dn: cn=user1,ou=users,o=ABCcompany
   objectclass: ABCperson
   ibm-group: group1
   ... other user attributes
   ```

3. Use static groups instead of dynamic groups for some of the dynamic groups that cannot be indexed.

Warning regarding DB2 logging of large static group updates

DB2 logging capacity might become a factor for updates to large static groups within TDBM. Each update to a large group is done in a single unit of work in DB2. A request to delete a large static group, or to delete many members from it, or to replace all of its members with a new list of members requires substantial DB2 logging space for the deleted rows associated with the data. For deletion of millions of members, this can potentially use thousands of cylinders of logging space in a single work unit. This can be problematic if a failure occurs during the update, especially if the beginning of the work unit extends back through many log data sets or beyond the oldest archived log data set. Large-scale deletion should be done piecemeal, modifying the group to delete members in blocks of a more manageable size (a few thousand at a time) until the entry is small enough to be deleted.

LE heap pools considerations

By default, the z/OS LDAP server uses LE heap pools 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 heap pools as compared to running without the facility enabled. Also, when storage is allocated to a given LE heap pool, it remains allocated to that heap pool and can only be used for future storage requests that are eligible (based on size) for the given heap pool. For example, when the z/OS LDAP server must process a large access group entry in storage, the following might occur:

- While the request is processing, the z/OS LDAP server might 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.
- Because of the sudden, large demand for storage to process the large group, most or all of the storage available to the z/OS LDAP server might be allocated and reserved to specific heap pools. Although the z/OS LDAP server might appear to be available and able to process various requests, many subsequent requests might fail because of 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 might be detectable by system resource monitoring products, such as the Resource Measurement Facility™ (RMF™).

If these problems occur, consider the following alternatives:

- Increase the z/OS LDAP server REGION (for 31-bit mode) or MEMLIMIT (for 64-bit mode).
- If running in 31-bit mode and REGION is already maximized, run in 64-bit mode with more storage.
- Tune the heap pool sizes to match your storage usage.
- Disable heap pools. Note that this alternative reduces the total heap storage requirements of the LDAP server, but significantly increases processor consumption.

Tuning LE heap and heap pools

By default, the z/OS LDAP server uses modest heap segment sizes so that typical installations are not forced to make adjustments. However, if you have large LDBM databases, it might be beneficial to increase the size of the primary and secondary heap segments to reduce the number of segments allocated. This can improve performance, and in some cases, reduce storage fragmentation.

The heap segment sizes are controlled by the LE runtime option 'HEAP' in 31-bit mode, or 'HEAP64' in 64-bit mode.

In 64-bit mode, the LDAP server uses most of its heap storage above the 2G bar. This is where LDBM data exists, and where tuning might help. A small amount of storage is also used above the 16M line and below the 2G bar, as required by some system services. The 31-bit addressable heap segments generally do not require tuning in 64-bit mode.

The z/OS LDAP server also uses heap pools to improve performance. The default settings are of modest size so that typical installations are not forced to make adjustments. However, the size and number of requests for heap storage varies by installation, according to the data placed in the directory databases. In particular, the data within large LDBM directories dominates the storage usage. Tuning the heap pools so that the cell sizes closely match the data might provide reduced storage consumption. Benefit can also be achieved by increasing the heap pool extent sizes to reduce the total number of heap pool extents allocated.

Overriding the heap pool settings for the LDAP server can be done by specifying the LE runtime option 'HEAPPOOLS' when running in 31-bit mode, or 'HEAPPOOLS64' in 64-bit mode. The 'HEAPPOOLS' option is also available in 64-bit mode for heap pool usage above the 16M line and below the 2G bar. However, the default setting for 31-bit addressable HEAPPOOLS in 64-bit mode usually does not need adjustment.

See *z/OS Language Environment Programming Guide* for details about how to tune the heap and heap pool settings. Note that the procedure for tuning these settings requires a controlled environment with representative workloads. Also, it is suggested that the storage reports needed for the tuning procedure are gathered in a non-production environment because tracking the storage statistics significantly affects performance.

Overrides for the LE runtime options can be specified in the PARM field on the EXEC statement or within a data set specified on a CEEOPTS DD statement in the LDAP server JCL. For more details, see *z/OS Language Environment Programming Reference* and *z/OS Language Environment Programming Guide*.

Paged search considerations

Paged searches allow an LDAP client to control the rate at which search entries are returned by requesting a page of entries at a time (the number of entries in a page is controlled by the client request). The LDAP client proceeds through the search result set by issuing successive requests for the next page of entries.

The LDAP server manages paged search requests in storage, caching the returned entries for the result set until they are retrieved by the client. Pervasive use of this capability by client applications increases the storage requirements of the LDAP server to hold these result sets. In particular, very large result sets use large amounts of storage for a paged search. If paged searches are enabled in the LDAP server, consider increasing the storage allowed in the LDAP server. If necessary, consider running the LDAP server in 64-bit addressing mode.

The following configuration options contribute in controlling the resources that might be used for paged search requests:

ibm-slapdPagedResAllowNonAdmin
: This is a configuration attribute in the **cn=configuration** entry in the CDBM backend. Setting this attribute value to "false" restricts paged search requests to administrators. This avoids widespread use of the facility and its associated resources on behalf of general users.

ibm-slapdPagedResLmt
: This is a configuration attribute in the **cn=configuration** entry in the CDBM backend. This limits the number of concurrent paged search requests on a given connection. For cases where LDAP client applications issue multiple active requests per connection, limiting this to a small number can help limit the overall number of paged search result sets that are cached in the LDAP server. Setting this to "0" disables paged searches.

sizeLimit
: This is a configuration option in the LDAP server configuration file that limits the number of entries returned on a given search request. This is also managed with group search limits. See “Managing group search

limits" on page 423 for more information about group search limits. Search size limits inherently limit the size of the paged search result sets cached by the LDAP server.

See "cn=configuration" on page 139 for more information about the **ibm-slapdPagedResLmt** and **ibm-slapdPagedResAllowNonAdmin** attributes in the **cn=configuration** entry. Also, see "pagedResults" on page 683 for more information about the **pagedResults** server control.

Sorted search considerations

Server-side sort allows an LDAP client to specify the order of returned search entries based on the values of attributes within the entries. Sorting the search results requires additional processor time in the LDAP server, and might also increase the storage used by the LDAP server. Generally, sorted searches use less than 5% of additional processor time compared to searches without sorting.

In many cases, sorting of results might be done in storage within the LDAP server. This might increase the overall storage requirements of the LDAP server. In particular, very large result sets use large amounts of storage for a sorted search. If sorted searches are enabled in the LDAP server, consider increasing the storage allowed in the LDAP server. If necessary, consider running the LDAP server in 64-bit addressing mode.

When possible, sorting might be done in DB2 for searches confined to the TDBM or DB2-based GDBM backends. DB2 sorting is done on the indexed VALUE column in the DIR_SEARCH table. In some cases, DB2 sorting might be attempted, but ultimately additional sorting might be required in storage depending on the data returned on the search.

DB2 sorting is not used when:

- the search results are found in the TDBM filter cache
- performing null-based subtree searches that search multiple backends
- alias dereferencing is requested and aliases exist within the TDBM backend
- reverse ordering is specified
- a sort key is an attribute whose syntax is octetString
- a sort key is an attribute eligible for encryption or hashing in the backend
- a sort key is an attribute whose syntax is distinguishedName, including **ibm-slapdDN**
- a sort key is an attribute whose syntax is generalizedTime or UTCTime and customization was done on the DIR_SEARCH table to shorten the VALUE column length to 20 bytes or less
- a sort key is an operational attribute not used within the searchable data of the entry. These are attributes that cannot be specified on search filters, such as **ibm-allGroups** or **entryOwner**.

DB2 sorting is used, but additional sorting is performed in storage when:

- read permission is denied on one or more of the sort key attributes for a returned entry
- a value for one of the sort key attributes is longer than the DIR_VALUE column in the DIR_SEARCH table (in this case, the DB2 ordering is not precise)

The following configuration options contribute in controlling the resources that might be used for sorted search requests:

ibm-slapdSortSrchAllowNonAdmin
This is a configuration attribute in the **cn=configuration** entry in the CDBM backend. Setting the attribute value to "false" restricts sorted search requests to administrators. This avoids widespread use of the facility and its associated resources on behalf of general users.

ibm-slapdSortKeyLimit
This is a configuration attribute in the **cn=configuration** entry in the CDBM backend. Setting the value of the attribute to "0" disables server-side sorting. This specifies the maximum number of sort keys that can be included on a single sorted search request.

See “cn=configuration” on page 139 for more information about the **ibm-slapdSortKeyLimit** and **ibm-slapdSortSrchAllowNonAdmin** attributes in the **cn=configuration** entry. Also, see “SortKeyRequest” on page 691 for more information about the **SortKeyRequest** server control.

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 needed.

GDBM can be configured to store its entries in DB2 (such as TDBM) or in files (such as LDBM). Most of the performance considerations for GDBM are identical to those of TDBM or LDBM, depending on which configuration you select. For example, if you select a DB2-based GDBM, DB2 tuning and cache tuning are important. If you select a file-based GDBM, storage, DASD space, initialization elapsed time, and database commit settings are important, especially if you allow the changelog to contain many entries. However, the following GDBM-specific differences should be noted as compared to TDBM or LDBM:

- The distinguished names (DNs) of entries and the searchable attributes within entries in GDBM tend to be well bounded in size and content. Therefore, when you configure GDBM to be DB2-based, 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 is disabled by default. You may enable the cache, if that is what you want, but if this is done, it is suggested that the cache is monitored to ensure that it is providing a benefit. Note that the entry cache is not implemented in file-based backends.
- When the **changeLogMaxAge** or **changeLogMaxEntries** option is specified in the GDBM section of the LDAP server configuration file, the change log 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 90.
- Since GDBM generally experiences high volumes of update activity, if you are using DB2-based GDBM, consider setting LOCKSIZE ROW on the table spaces that contain the **DIR_ENTRY** and the **DIR_SEARCH** table to help avoid DB2 deadlocks.

SDBM performance considerations

The z/OS LDAP server SDBM backend allows access to the RACF database. Most tuning that affects performance in this area is within the RACF product. See *z/OS MVS Initialization and Tuning Guide* for more information about tuning RACF.

Also, see "SDBM operational behavior" on page 342 for details about different types of LDAP requests supported, and the RACF operations issued by these requests. This information can also be helpful when assessing RACF tuning considerations.

When writing applications that only require authentication to the SDBM backend by using LDAP bind requests, performance can be improved by specifying the **authenticateOnly** control on the bind request within the application. See "authenticateOnly" on page 679 for more information.

Part 3. Appendixes

Appendix A. Initial LDAP server schema

This topic shows the initial schema established when the LDAP server is first started. The initial schema is always part of the LDAP server schema and the elements in the initial schema cannot be deleted. With several exceptions, the initial schema cannot be modified. See “Updating the schema” on page 297 for more information.

```
cn=schema
objectclass=ibmSubschema
objectclass=subentry
objectclass=subschema
objectclass=top
subtreespecification=NULL
ldapsyntaxes=( 1.3.18.0.2.8.1 DESC 'IBM attribute type description' )
ldapsyntaxes=( 1.3.18.0.2.8.3 DESC 'IBM entry UUID' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.10 DESC 'Certificate pair' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.11 DESC 'Country string' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.12 DESC 'Distinguished name' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.14 DESC 'Delivery method' )
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=( 1.3.6.1.4.1.1466.115.121.1.21 DESC 'Enhanced guide' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.22 DESC 'Facsimile telephone number' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.23 DESC 'Fax' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.24 DESC 'Generalized time' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.25 DESC 'Guide' )
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=( 1.3.6.1.4.1.1466.115.121.1.28 DESC 'JPEG' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.3 DESC 'Attribute type description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.30 DESC 'Matching rule description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.31 DESC 'Matching rule use description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.33 DESC 'MHS OR address' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.34 DESC 'Name and optional UID' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.35 DESC 'Name form description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.36 DESC 'Numeric string' )
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 'Object identifier' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.39 DESC 'Other mailbox' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.40 DESC 'Octet string' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.41 DESC 'Postal address' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.42 DESC 'Protocol information' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.43 DESC 'Presentation address' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.44 DESC 'Printable string' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.49 DESC 'Supported algorithm' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.5 DESC 'Binary' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.50 DESC 'Telephone number' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.51 DESC 'Teletex terminal identifier' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.52 DESC 'Telex number' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.53 DESC 'UTC time' )
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=( 1.3.6.1.4.1.1466.115.121.1.6 DESC 'Bit string' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.7 DESC 'Boolean' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.8 DESC 'Certificate' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.9 DESC 'Certificate list' )
matchingrules=( 1.3.18.0.2.22.2 NAME ( 'ibm-entryUuidMatch' ) SYNTAX 1.3.18.0.2.8.3 )
matchingrules=( 1.3.18.0.2.4.405 NAME ( 'distinguishedNameOrderingMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
matchingrules=( 1.3.6.1.4.1.1466.109.114.1 NAME ( 'caseExactIA5Match' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
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=( 1.3.6.1.4.1.1466.109.114.3 NAME ( 'caseIgnoreIA5SubstringsMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
matchingrules=( 2.5.13.0 NAME ( 'objectIdentifierMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )
matchingrules=( 2.5.13.1 NAME ( 'distinguishedNameMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
matchingrules=( 2.5.13.10 NAME ( 'numericStringSubstringsMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.36 )
matchingrules=( 2.5.13.11 NAME ( 'caseIgnoreListMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.41 )
matchingrules=( 2.5.13.12 NAME ( 'caseIgnoreListSubstringsMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.41 )
matchingrules=( 2.5.13.13 NAME ( 'booleanMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )
matchingrules=( 2.5.13.14 NAME ( 'integerMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )
matchingrules=( 2.5.13.15 NAME ( 'integerOrderingMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )
matchingrules=( 2.5.13.16 NAME ( 'bitStringMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.6 )
matchingrules=( 2.5.13.17 NAME ( 'octetStringMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 )
matchingrules=( 2.5.13.18 NAME ( 'octetStringOrderingMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 )
matchingrules=( 2.5.13.2 NAME ( 'caseIgnoreMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
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 1.3.6.1.4.1.1466.115.121.1.50 )
matchingrules=( 2.5.13.22 NAME ( 'presentationAddressMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.43 )
matchingrules=( 2.5.13.23 NAME ( 'uniqueMemberMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.34 )
matchingrules=( 2.5.13.24 NAME ( 'protocolInformationMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.42 )
matchingrules=( 2.5.13.25 NAME ( 'utcTimeMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.53 )
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=( 2.5.13.29 NAME ( 'integerFirstComponentMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )
matchingrules=( 2.5.13.3 NAME ( 'caseIgnoreOrderingMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
matchingrules=( 2.5.13.30 NAME ( 'objectIdentifierFirstComponentMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )
matchingrules=( 2.5.13.34 NAME ( 'certificateExactMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.8 )
matchingrules=( 2.5.13.35 NAME ( 'certificateMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.8 )
matchingrules=( 2.5.13.4 NAME ( 'caseIgnoreSubstringsMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
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 1.3.6.1.4.1.1466.115.121.1.15 )
matchingrules=( 2.5.13.8 NAME ( 'numericStringMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.36 )
matchingrules=( 2.5.13.9 NAME ( 'numericStringOrderingMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.36 )
attributetypes=( 0.9.2342.19200300.100.1.1 NAME ( 'uid' ) DESC 'User shortname or userid'
```

```
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 0.9.2342.19200300.100.1.23 NAME ( 'lastmodifiedtime' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 USAGE userApplications )
attributetypes=( 0.9.2342.19200300.100.1.24 NAME ( 'lastmodifiedby' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userApplications )
attributetypes=( 1.2.840.113556.1.4.656 NAME ( 'userPrincipalName' )
 DESC 'Primary security identity in the form <principal>@<realm>' EQUALITY caseExactMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.2.840.113556.1.4.77 NAME ( 'maxTicketAge' )
 DESC 'Value defining the maximum lifetime of a user ticket'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.2.840.113556.1.4.867 NAME ( 'altSecurityIdentities' )
 DESC 'Alternate security identities' EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1068 NAME ( 'ibm-kn' 'ibm-kerberosName' )
 DESC 'Access control list definition for a Kerberos identity in the format <principal>@<realm>'
 EQUALITY caseExactMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1088 NAME ( 'krbAliasedObjectName' )
 DESC 'Contains the DN of the aliased object' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1091 NAME ( 'krbPrincipalName' )
 DESC 'Kerberos principal name in the format <princ-name>@<realm-name>'
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1099 NAME ( 'racfLNotesShortName' )
 DESC 'represents the SNAME field of the RACF LNOTES segment'
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1100 NAME ( 'racfNDSUserName' )
 DESC 'Represents the UNAME field of the RACF NDS segment' EQUALITY caseExactMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 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 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1146 NAME ( 'racfConnectCount' ) DESC 'RACF Connect Count'
 EQUALITY caseIgnoreIA5Match SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1147 NAME ( 'racfConnectLastConnect' ) DESC 'RACF Connect Last Connect'
 EQUALITY caseIgnoreIA5Match SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1148 NAME ( 'racfConnectOwner' ) DESC 'RACF Connect Owner'
 EQUALITY caseIgnoreIA5Match SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1149 NAME ( 'racfConnectResumeDate' ) DESC 'RACF Connect Resume Date'
 EQUALITY caseIgnoreIA5Match SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1150 NAME ( 'racfConnectRevokeDate' ) DESC 'RACF Connect Revoke Date'
 EQUALITY caseIgnoreIA5Match SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1151 NAME ( 'racfGroupId' ) DESC 'RACF group ID' EQUALITY caseIgnoreIA5Match
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1152 NAME ( 'racfUserid' ) DESC 'RACF userid' EQUALITY caseIgnoreIA5Match
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1153 NAME ( 'racfCurKeyVersion' ) DESC 'Current key version'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1154 NAME ( 'krbHintAliases' ) DESC 'Entries that can be associated with this entry'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1155 NAME ( 'ibm-changeInitiatorsName' ) DESC 'The DN of the entity that initiated the change'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1156 NAME ( 'krbPrincSubtree' ) DESC 'List of DNs under which principals in this realm reside'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1157 NAME ( 'krbRealmName-V2' ) DESC 'Kerberos realm name' EQUALITY caseExactMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1158 NAME ( 'ibm-nativeId' ) DESC 'Userid in the native security manager'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1162 NAME ( 'racfLDAPBindDN' ) DESC 'RACF LDAP Bind DN' EQUALITY caseExactMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1163 NAME ( 'racfLDAPBindPw' ) DESC 'RACF LDAP Bind Password' EQUALITY caseExactMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1164 NAME ( 'racfLDAPHost' ) DESC 'RACF LDAP Host' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.155 NAME ( 'secretKey' ) DESC 'Attribute is always stored in encrypted form'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1780 NAME ( 'ibm-EntryUUID' ) DESC 'Uniquely identifies an LDAP entry throughout its life'
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.18.0.2.8.3 USAGE dSAOperation )
attributetypes=( 1.3.18.0.2.4.185 NAME ( 'sysplex' ) DESC 'Identifies the name of a z/OS sysplex'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.186 NAME ( 'profileType' ) DESC 'Identifies the name of a z/OS Security Server profile'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.187 NAME ( 'racfid' ) DESC 'Identifies the name of a z/OS Security Server userid or groupid'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.188 NAME ( 'racfAuthorizationDate' ) DESC 'Date is displayed in yy.ddd format'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.189 NAME ( 'racfOwner' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.190 NAME ( 'racfInstallationData' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.191 NAME ( 'racfDatasetModel' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.1913 NAME ( 'racfGroupUniversal' ) DESC 'RACF universal group indicator'
 EQUALITY caseIgnoreIA5Match SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.192 NAME ( 'racfSuperiorGroup' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.193 NAME ( 'racfGroupNoTermUAC' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.194 NAME ( 'racfSubGroupName' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.195 NAME ( 'racfGroupUserids' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.197 NAME ( 'racfAttributes' )
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.198 NAME ( 'racfPassword' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.199 NAME ( 'racfPasswordInterval' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.200 NAME ( 'racfPasswordChangeDate' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2007 NAME ( 'racfEncryptType' ) DESC 'RACF encrypt type'
 EQUALITY caseIgnoreIA5Match  SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.201 NAME ( 'racfProgrammerName' )
```

```
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.202 NAME ( 'racfDefaultGroup' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.203 NAME ( 'racfLastAccess' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.204 NAME ( 'racfSecurityLevel' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.205 NAME ( 'racfSecurityCategoryList' )
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.206 NAME ( 'racfRevokeDate' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.207 NAME ( 'racfResumeDate' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.208 NAME ( 'racfLogonDays' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.209 NAME ( 'racfLogonTime' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.210 NAME ( 'racfClassName' )
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.211 NAME ( 'racfConnectGroupName' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.212 NAME ( 'racfConnectGroupAuthority' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.213 NAME ( 'racfConnectGroupUACC' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.214 NAME ( 'racfSecurityLabel' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.215 NAME ( 'SAFDfpDataApplication' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.216 NAME ( 'SAFDfpDataClass' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.217 NAME ( 'SAFDfpManagementClass' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.218 NAME ( 'SAFDfpStorageClass' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.219 NAME ( 'racfOmvsGroupId' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.220 NAME ( 'racfOvmGroupId' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.221 NAME ( 'SAFAccountNumber' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.222 NAME ( 'SAFDefaultCommand' )
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.223 NAME ( 'SAFDestination' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2239 NAME ( 'racfLDAPProf' )
 DESC 'RACF LDAP Profile Name' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.224 NAME ( 'SAFHoldClass' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2240 NAME ( 'racfOmvsGroupIdKeyword' )
 DESC 'RACF group OMVS keyword' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2241 NAME ( 'racfOmvsUidKeyword' )
 DESC 'RACF user OMVS keyword' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2242 NAME ( 'ibm-memberGroup' )
 DESC 'Identifies subgroups of a parent group' EQUALITY distinguishedNameMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2243 NAME ( 'ibm-allMembers' )
 DESC 'Lists all members of a group' NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.2244 NAME ( 'ibm-allGroups' )
 DESC 'Lists all groups containing an entry' NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.225 NAME ( 'SAFJobClass' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.226 NAME ( 'SAFMessageClass' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.227 NAME ( 'SAFDefaultLoginProc' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.228 NAME ( 'SAFLogonSize' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.229 NAME ( 'SAFMaximumRegionSize' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.230 NAME ( 'SAFDefaultSysoutClass' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.231 NAME ( 'SAFUserdata' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.232 NAME ( 'SAFDefaultUnit' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2325 NAME ( 'ibm-entryChecksum' )
 DESC 'A checksum of the user attributes for the entry containing this attribute.'
 EQUALITY caseExactIA5Match SINGLE-VALUE NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.2326 NAME ( 'ibm-entryChecksumOp' )
 DESC 'A checksum of the replicated operational attributes for the entry containing this attribute.'
 EQUALITY caseExactIA5Match SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.2327 NAME ( 'ibm-supportedReplicationModels' )
 DESC 'Advertises in the Root DSE the OIDs of replication models supported by the server'
 EQUALITY caseExactIA5Match NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE dSAOperation )
attributetypes=( 1.3.18.0.2.4.2328 NAME ( 'ibm-serverId' )
 DESC 'Advertises in the Root DSE the ibm-slapdServerId configuration setting'
 EQUALITY caseExactIA5Match SINGLE-VALUE NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE dSAOperation )
attributetypes=( 1.3.18.0.2.4.2329 NAME ( 'ibm-replicationServerIsMaster' )
 DESC 'Indicates that a server assumes the role of a master for a given subtree'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.233 NAME ( 'SAFTsoSecurityLabel' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2330 NAME ( 'ibm-replicationChangeLDIF' )
 DESC 'Provides LDIF representation of the last failing operation'
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.2331 NAME ( 'ibm-effectiveReplicationModel' )
```

```
 DESC 'Advertises in the Root DSE the OID of the replication model in use by the server'
 EQUALITY caseExactIA5Match SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.2332 NAME ( 'ibm-replicationLastResultAdditional' )
 DESC 'Provides any additional error information returned by the consuming server in the message
 component of the LDAP result' EQUALITY caseIgnoreMatch NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.2333 NAME ( 'ibm-replicationPendingChangeCount' )
 DESC 'Indicates the total number of pending unreplicated changes for this replication agreement'
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.2334 NAME ( 'ibm-replicationLastChangeId' )
 DESC 'Indicates last change id successfully replicated for a replication agreement'
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.2335 NAME ( 'ibm-replicationLastFinishTime' )
 DESC 'Indicates the last time the replication thread completed sending all of the pending entries.'
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.2336 NAME ( 'ibm-replicationState' )
 DESC 'Indicates the state of the replication thread: active,ready,waiting,suspended, or full;
 if full, the value will indicate the amount of progress' EQUALITY caseExactIA5Match
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.2337 NAME ( 'ibm-replicationPendingChanges' )
 DESC 'Unreplicated change in the form <change id> <operation> <dn> where operation
 is ADD, DELETE, MODIFY, MODIFYDN' EQUALITY caseIgnoreMatch NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.2338 NAME ( 'ibm-replicationLastActivationTime' )
 DESC 'Indicates the last time the replication thread was activated'
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.2339 NAME ( 'ibm-replicationNextTime' )
 DESC 'Indicates next scheduled time for replication' SINGLE-VALUE NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.234 NAME ( 'racfPrimaryLanguage' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2340 NAME ( 'ibm-replicationLastResult' )
 DESC 'Result of last attempted replication in the form: <time> <change-id> <result code>
 <operation> <entry-dn>' EQUALITY caseIgnoreMatch SINGLE-VALUE NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.2341 NAME ( 'ibm-replicationBatchStart' )
 DESC 'Time to replicate accumulated changes in the form of Thhmmss where hh is
 hours, mm is minutes and ss is seconds, using a 24 hour clock' EQUALITY caseExactIA5Match
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2342 NAME ( 'ibm-replicaKeyfile' )
 DESC 'Name of key database file on the supplying server with the certificate of the
 consuming server and the supplier' EQUALITY caseIgnoreMatch SINGLE-VALUE
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2343 NAME ( 'ibm-replicaKeylabel' )
 DESC 'Label for certificate containing private key for supplying server'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2344 NAME ( 'ibm-scheduleTuesday' )
 DESC 'DN of the entry defining the replication schedule for Tuesday'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2345 NAME ( 'ibm-replicationTimesUTC' )
 DESC 'Scheduled times are GMT if TRUE or local time zone if FALSE'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2346 NAME ( 'ibm-scheduleFriday' )
 DESC 'DN of the entry defining the replication schedule for Friday'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2347 NAME ( 'ibm-scheduleSaturday' )
 DESC 'DN of the entry defining the replication schedule for Saturday'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2348 NAME ( 'ibm-scheduleWednesday' )
 DESC 'DN of the entry defining the replication schedule for Wednesday'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2349 NAME ( 'ibm-replicationOnHold' )
 DESC 'Indicates replication is suspended when TRUE'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.235 NAME ( 'racfSecondaryLanguage' )
 DESC 'Secondary language' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2350 NAME ( 'ibm-scheduleSunday' )
 DESC 'DN of the entry defining the replication schedule for Sunday'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2351 NAME ( 'ibm-replicaCredentialsDN' )
 DESC 'DN of the entry containing the credentials for the replication agreement'
 EQUALITY distinguishedNameMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2352 NAME ( 'ibm-replicationImmediateStart' )
 DESC 'Time to start replicating changes as they occur and ended by next ibm-replicationBatchStart
 time in the form of Thhmmss where hh is hours, mm is minutes and ss is seconds,
 using a 24 hour clock' EQUALITY caseExactIA5Match SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2353 NAME ( 'ibm-scheduleMonday' )
 DESC 'DN of the entry defining the replication schedule for Monday'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2354 NAME ( 'ibm-replicaKeypwd' )
 DESC 'Password for file named by ibm-replicaKeyfile' SINGLE-VALUE
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2355 NAME ( 'ibm-replicaScheduleDN' )
 DESC 'DN of the entry containing the schedule for the replication agreement'
 EQUALITY distinguishedNameMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2356 NAME ( 'ibm-scheduleThursday' ) D
 ESC 'DN of the entry defining the replication schedule for Thursday'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2357 NAME ( 'ibm-replicaConsumerId' )
 DESC 'Specifies the server ID of the server that is supplied by a replication agreement'
 EQUALITY caseExactIA5Match SUBSTR caseExactSubstringsMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2358 NAME ( 'ibm-replicaReferralURL' )
 DESC 'Ordered list of LDAP URLs with server name and optional port numbers, e.g.,
 ldap://host:port separated by spaces' EQUALITY caseIgnoreMatch SUBSTR
 caseIgnoreSubstringsMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2359 NAME ( 'ibm-replicaServerId' )
 DESC 'Identifies the server acting as supplier for a set of replicas'
 EQUALITY caseExactIA5Match SUBSTR caseExactSubstringsMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.236 NAME ( 'racfOperatorIdentification' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2360 NAME ( 'ibm-replicaURL' )
```

```
 DESC 'Specifies the LDAP URL that should be used to contact the consumer
 server during replication, e.g., ldap[s]://host[:port]' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2361 NAME ( 'ibm-replicaGroup' )
 DESC 'Indicates the name of the entry containing the collection of servers
 participating in replication' SUP name SINGLE-VALUE USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2367 NAME ( 'ibm-slapdReplicaSubtree' )
 DESC 'A DN identifying the top of a replicated subtree.'
 EQUALITY distinguishedNameMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.237 NAME ( 'racfOperatorClass' )
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2376 NAME ( 'ibm-slapdPagedResAllowNonAdmin' )
 DESC 'Whether or not the server should allow non-Administrators to request paged search results.'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.2377 NAME ( 'ibm-slapdSortSrchAllowNonAdmin' )
 DESC 'Whether or not the server should allow non-Administrators to request sorted search results.'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.238 NAME ( 'racfOperatorPriority' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2380 NAME ( 'ibm-slapdPagedResLmt' )
 DESC 'Maximum number of outstanding paged search requests allowed simultaneously.'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.2381 NAME ( 'ibm-slapdSortKeyLimit' )
 DESC 'Maximum number of sort keys that can be specified on a single sorted search request.'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.239 NAME ( 'racfOperatorReSignon' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.240 NAME ( 'racfTerminalTimeout' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2401 NAME ( 'ibm-slapdMasterReferral' )
 DESC 'URL of master replica server (e.g.: ldaps://master.us.ibm.com:636)'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.2409 NAME ( 'ibm-slapdMasterDN' )
 DESC 'Bind DN used by a replication supplier server. The value has to match
 the replicaBindDN in the credentials object associated with the replication agreement.
 When kerberos is used to authenticate to the replica, ibm-slapdMasterDN must specify the
 DN representation of the kerberos ID (e.g. ibm-kn=freddy@realm1). When kerberos is used,
 MasterServerPW is ignored.' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.241 NAME ( 'racfStorageKeyword' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2411 NAME ( 'ibm-slapdMasterPW' )
 DESC 'Bind password used by replication supplier server. The value has to
 match the replicaBindPW in the credentials object associated with the
 replication agreement. When kerberos is used, MasterServerPW is ignored.'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.242 NAME ( 'racfAuthKeyword' )
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2420 NAME ( 'ibm-slapdKrbAdminDN' )
 DESC 'Specifies the kerberos ID of the LDAP administrator (e.g. ibm-kn=name@realm).
 Used when kerberos authentication is used to authenticate the administrator when
 logged onto the Web Admin interface. This is specified instead of adminDN and adminPW.'
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.2425 NAME ( 'ibm-slapdAdminPW' ) DESC 'Bind
 password for ibmslapd administrator.' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.2428 NAME ( 'ibm-slapdAdminDN' )
 DESC 'Bind DN for ibmslapd administrator, e.g.: cn=root' EQUALITY distinguishedNameMatch
 ORDERING distinguishedNameOrderingMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.243 NAME ( 'racfMformKeyword' )
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2430 NAME ( 'ibm-slapdInvalidLine' )
 DESC 'This attribute will be prepended to the beginning of any configuration attribute
 for which the value is invalid. This allows invalid configuration settings to be
 identified with a simple search for ibm-slapdInvalidLine=*.' EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2433 NAME ( 'ibm-slapdServerId' )
 DESC 'Identifies the server for use in replication'
 EQUALITY caseExactIA5Match SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.244 NAME ( 'racfLevelKeyword' )
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2449 NAME ( 'ibm-slapdDN' )
 DESC 'This attribute is used to sort search results by the entry DN'
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.245 NAME ( 'racfMonitorKeyword' )
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.246 NAME ( 'racfRoutcodeKeyword' )
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.247 NAME ( 'racfLogCommandResponseKeyword' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.248 NAME ( 'racfMGIDKeyword' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2481 NAME ( 'ibm-supportedCapabilities' )
 DESC 'Capabilities supported by this server' NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE dSAOperation )
attributetypes=( 1.3.18.0.2.4.2482 NAME ( 'ibm-enabledCapabilities' )
 DESC 'Capabilities that are enabled for use on this server' NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE dSAOperation )
attributetypes=( 1.3.18.0.2.4.2484 NAME ( 'ibm-replicationExcludedCapability' )
 DESC 'The values are OIDs associated with server capabilities.  Objects and
 attributes related to the specified capabilities will not be replicated under
 the agreement containing this attribute.' EQUALITY caseIgnoreMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2486 NAME ( 'ibm-slapdMaxPendingChangesDisplayed' )
DESC 'Maximum number of pending replication updates to be displayed for any given
 replication agreement on a supplier server.' SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.249 NAME ( 'racfDOMKeyword' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2493 NAME ( 'ibm-pwdPolicy' )
 DESC 'Specifies with a value of TRUE that Password Policy is turned on.'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2494 NAME ( 'ibm-replDailySchedName' )
 DESC 'Naming attribute and descriptive name for an ibm-replicaDailySchedule object.'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
```

```
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2495 NAME ( 'ibm-replicationThisServerIsMaster' )
 DESC 'Indicates whether the server returning this attribute is a master server
 for the subtree containing this entry.' SINGLE-VALUE NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE dSAOperation )
attributetypes=( 1.3.18.0.2.4.2496 NAME ( 'ibm-replCredName' )
 DESC 'Naming attribute and descriptive name for an ibm-replicaCredentials object.'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2497 NAME ( 'ibm-replWeeklySchedName' )
 DESC 'Naming attribute and descriptive name for an ibm-replicaWeeklySchedule object.'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2498 NAME ( 'ibm-replicationIsQuiesced' )
 DESC 'Indicates whether the replicated subtree containing this attribute is quiesced on this server.'
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE dSAOperation )
attributetypes=( 1.3.18.0.2.4.250 NAME ( 'racfKEYKeyword' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.2500 NAME ( 'ibm-slapdMigrationInfo' )
 DESC 'Information used to control migration of a component.'
 EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.251 NAME ( 'racfCMDSYSKeyword' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.252 NAME ( 'racfUDKeyword' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.253 NAME ( 'racfMscopeSystems' )
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.254 NAME ( 'racfAltGroupKeyword' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.255 NAME ( 'racfAutoKeyword' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.256 NAME ( 'racfWorkAttrUsername' )
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.257 NAME ( 'racfBuilding' )
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.258 NAME ( 'racfDepartment' )
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.259 NAME ( 'racfRoom' )
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.260 NAME ( 'racfWorkAttrAccountNumber' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.261 NAME ( 'racfAddressLine1' )
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.262 NAME ( 'racfAddressLine2' )
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.263 NAME ( 'racfAddressLine3' )
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.264 NAME ( 'racfAddressLine4' )
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.265 NAME ( 'racfOmvsUid' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.266 NAME ( 'racfOmvsHome' )
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.267 NAME ( 'racfOmvsInitialProgram' )
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.268 NAME ( 'racfNetviewInitialCommand' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.269 NAME ( 'racfDefaultConsoleName' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.270 NAME ( 'racfCTLKeyword' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.271 NAME ( 'racfMSGRCVRKeyword' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.272 NAME ( 'racfNetviewOperatorClass' )
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.273 NAME ( 'racfDomains' )
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.274 NAME ( 'racfNGMFADMKeyword' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.275 NAME ( 'racfDCEUUID' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.276 NAME ( 'racfDCEPrincipal' )
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.277 NAME ( 'racfDCEHomeCell' )
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.278 NAME ( 'racfDCEHomeCellUUID' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.279 NAME ( 'racfDCEAutoLogin' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.280 NAME ( 'racfOvmUid' )
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.281 NAME ( 'racfOvmHome' )
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.282 NAME ( 'racfOvmInitialProgram' )
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.283 NAME ( 'racfOvmFileSystemRoot' )
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.285 NAME ( 'aclEntry' )
 DESC 'Defines an access list entry' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.286 NAME ( 'aclPropagate' )
 DESC 'Defines access list subtree propagation'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.287 NAME ( 'aclSource' )
 DESC 'Source of the access list for an entry'
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.288 NAME ( 'entryOwner' )
 DESC 'Defines an entry owner' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.289 NAME ( 'ownerPropagate' )
 DESC 'Defines entry owner subtree propagation'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.290 NAME ( 'ownerSource' )
 DESC 'Source of the owner for an entry' SINGLE-VALUE NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.298 NAME ( 'replicaHost' )
 DESC 'Specifies the replica host name' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.299 NAME ( 'replicaBindDN' )
```

```
 DESC 'Specifies the replica bind DN' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.300 NAME ( 'replicaCredentials' 'replicaBindCredentials' )
 DESC 'Specifies the replica bind credentials' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.301 NAME ( 'replicaPort' ) DESC 'Specifies the replica bind port'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.3013 NAME ( 'ibm-slapdAdminGroupEnabled' )
 DESC 'Must be one of { TRUE | FALSE }.  Specifies whether the Administrative Group is currently enabled.
 Defaults to FALSE if unspecified. If set to TRUE, the server will allow users in the administrative
 group to login.' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.302 NAME ( 'replicaBindMethod' )
 DESC 'Specifies the replica bind method' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.303 NAME ( 'replicaUseSSL' )
 DESC 'Specifies SSL usage when binding to replica' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.3032 NAME ( 'ibm-slapdDigestAdminUser' )
 DESC 'Specifies the Digest MD5 User Name of the LDAP administrator or administrative group member.
 Used when MD5 Digest authentication is used to authenticate an administrator.' EQUALITY caseExactMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.304 NAME ( 'replicaUpdateTimeInterval' )
 DESC 'Specifies replication update interval' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.3072 NAME ( 'ibm-searchSizeLimit' ) DESC 'Maximum number of entries
 to return from search requests for a member in a special search limit group. 0 = unlimited. -1 =
 ignored.' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.3073 NAME ( 'ibm-searchTimeLimit' ) DESC 'Maximum number of seconds
 to spend on search requests for a member in a special search limit group. 0 = unlimited. -1 = ignored.'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.3081 NAME ( 'ibm-saslDigestRealmName' )
 DESC 'DIGEST-MD5 realm names for this server' NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 USAGE dSAOperation )
attributetypes=( 1.3.18.0.2.4.3089 NAME ( 'racfOmvsSharedMemoryMaximum' )
 DESC 'Represents the SHMEMMAX(shared-memory-size) field of the RACF user OMVS segment'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 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'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3091 NAME ( 'racfPasswordEnvelope' ) DESC 'Envelope containing
 user password information' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3094 NAME ( 'firstChangeNumber' )
 DESC 'Change number for the earliest entry in the server change log' EQUALITY integerMatch
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE dSAOperation )
attributetypes=( 1.3.18.0.2.4.3095 NAME ( 'lastChangeNumber' )
 DESC 'Change number for the latest entry in the server change log' EQUALITY integerMatch
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE dSAOperation )
attributetypes=( 1.3.18.0.2.4.3097 NAME ( 'ldapServiceName' )
 DESC 'LDAP service name for this server as host@realm' SINGLE-VALUE NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE dSAOperation )
attributetypes=( 1.3.18.0.2.4.3098 NAME ( 'ibmDirectoryVersion' )
 DESC 'Version of this directory server' SINGLE-VALUE NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE dSAOperation )
attributetypes=( 1.3.18.0.2.4.3128 NAME ( 'ibm-slapdLog' ) DESC 'Log path and file name.
 On Windows, forward slashes are allowed, and a leading slash not preceded by a drive letter
 is assumed to be rooted at the install directory (i.e.: /tmp/bulkload.errors = D:\Program
 Files\IBM\ldap\tmp\bulkload.errors).' EQUALITY caseExactMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3129 NAME ( 'ibm-slapdLogMaxArchives' )
 DESC 'The maximum number of archived logs where 0 means no archive file will be kept and -1
 means an unlimited number of archive files will be kept.' EQUALITY integerMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3130 NAME ( 'ibm-slapdLogOptions' )
 DESC 'Any log options that the log uses, for example, log level or mask.' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3131 NAME ( 'ibm-slapdLogSizeThreshold' )
 DESC 'When this size threshold, in MB, is exceeded the file will be archived where 0
 means no threshold and thus no archiving.' EQUALITY integerMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3134 NAME ( 'ibm-slapdLogArchivePath' )
 DESC 'Path for archived files.  On Windows, forward slashes are allowed, and a
 leading slash not preceded by a drive letter is assumed to be rooted at the install
 directory (i.e.: /tmp = D:\Program Files\IBM\ldap\tmp).' EQUALITY caseExactMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3138 NAME ( 'ibm-replicationFailedChangeCount' )
 DESC 'Indicates the number of changes logged as failures for this replication agreement.'
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.3139 NAME ( 'ibm-replicationFailedChanges' )
 DESC 'Unreplicated change for a specific replication agreement in the form <change id>
 <operation> <dn> <LDAP result code> <timestamp> <failed attempts> where operation is ADD,
 DELETE, MODIFY or MODIFYDN.' EQUALITY caseIgnoreMatch NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.3141 NAME ( 'ibm-pwdAccountLocked' )
 DESC 'The indication that the users account has been locked'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.3142 NAME ( 'ibm-slapdReplConflictMaxEntrySize' )
 DESC 'Maximum number of bytes that an entry can contain and still be resent to a
 target server as a result of replication conflict resolution.  This value is dynamic.'
 EQUALITY integerMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3150 NAME ( 'ibm-replicaMethod' ) DESC 'Method used
 by a server to replicate 1=single thread, 2=multiple threads and connections.
 The value is not dynamic.' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3151 NAME ( 'ibm-replicaConsumerConnections' )
 DESC 'Specifies the number of LDAP connections to the consumer server during replication'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3152 NAME ( 'ibm-slapdReplMaxErrors' ) DESC 'Limit to
 allowed errors per replication agreement, -1=unlimited, 0=stop on error. The value is dynamic.'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3153 NAME ( 'ibm-slapdReplContextCacheSize' )
 DESC 'Maximum number of updates to retain in  replication context cache. The value is dynamic.'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 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.'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 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.'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
```

```
attributetypes=( 1.3.18.0.2.4.3223 NAME ( 'ibm-replicationperformance' )
 DESC 'Values for the following metrics: connection,operations queued,dependent operations
 queued,operations sent, dependent operations sent, operations received, errors'
 EQUALITY caseIgnoreMatch NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.3238 NAME ( 'ibm-pwdPolicyStartTime' )
 DESC 'Specifies the time Password Policy was last turned on' ORDERING generalizedTimeOrderingMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3239 NAME ( 'racfHcKeyword' ) DESC 'Represents the HC field of
 the RACF user OPERPARM segment' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3240 NAME ( 'racfNGMFVSPNKeyword' ) DESC 'Represents the
 NGMFVSPN field of the RACF user NETVIEW segment' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3241 NAME ( 'racfIntidsKeyword' ) DESC 'Represents the INTIDS
 field of the RACF user OPERPARM segment' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3242 NAME ( 'racfPassPhrase' ) DESC 'Represents the passphrase
 field of the RACF user base segment' EQUALITY caseExactMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3243 NAME ( 'racfUnknidsKeyword' )
 DESC 'Represents the UNKNIDS field of the RACF user OPERPARM segment'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3244 NAME ( 'racfHavePasswordEnvelope' )
 DESC 'Represents the password-enveloped field of the RACF user base segment'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3245 NAME ( 'racfPassPhraseChangeDate' )
 DESC 'Represents the last change date of the passphrase field of the RACF user base segment'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3261 NAME ( 'ibm-slapdLogCARSOptions' )
 DESC 'Any log options that the event formatted data sent to CARS uses, for example,
 log level or mask.' EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3263 NAME ( 'ibm-slapdLogCARSPort' )
 DESC 'The CARS servers port where the event formatted data will be sent. ' EQUALITY integerMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3264 NAME ( 'ibm-slapdLogEventFileOptions' )
 DESC 'Any log options that the event formatted log uses, for example, log level or mask.'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3265 NAME ( 'ibm-slapdLogCARSServer' ) DESC 'The CARS servers
 hostname where the event formatted data will be sent.' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3266 NAME ( 'ibm-slapdLogEventFileSizeThreshold' )
 DESC 'When this size threshold, in MB, is exceeded the event formatted file will be
 archived where 0 means no threshold and thus no archiving.' EQUALITY integerMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3267 NAME ( 'ibm-slapdLogEventFileMaxArchives' )
 DESC 'The maximum number of archived logs where 0 means no archive file will be
 kept and -1 means an unlimited number of archive files will be kept.' EQUALITY integerMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3268 NAME ( 'ibm-slapdLogEventFileArchivePath' )
 DESC 'Path for archived event formatted files. On Windows, forward slashes are allowed,
 and a leading slash not preceded by a drive letter is assumed to be rooted at the install
 directory (i.e.: /tmp = D:\Program Files\IBM\ldap\V6.1\tmp).' EQUALITY caseExactMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3269 NAME ( 'ibm-slapdLogCARSEnabled' ) DESC 'Must be one
 of [TRUE|FALSE]. Specifies whether the log data will be written to a CARS Server.'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3270 NAME ( 'ibm-slapdLogEventFileEnabled' )
 DESC 'Must be one of [TRUE|FALSE]. Specifies whether the log data will be written to
 event formatted log files.' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3288 NAME ( 'ibm-replicaPKCS11Enabled' )
 DESC 'Must be one of { TRUE | FALSE }. Specify whether PKCS11 interface is
 enable to do cryptographic operation and key database file lookup from installed
 crypto device.' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.3294 NAME ( 'ibm-replicationCreateMissingEntries' )
 DESC 'Indicates whether missing parent entries are to be created on consumer'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3295 NAME ( 'ibm-replicationFilterDN' )
 DESC 'A DN identifying the filter entry' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3296 NAME ( 'ibm-replicationFilterAttr' )
 DESC 'This attribute is used to hold the actual filter' EQUALITY caseIgnoreMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3299 NAME ( 'ibm-slapdAdminRole' )
 DESC 'Administrative role(s) associated with a Local Administrative Group Member.
 Valid values are AuditAdmin, DirDataAdmin, NoAdmin, PasswordAdmin, ReplicationAdmin,
 SchemaAdmin, ServerConfigGroupMember, ServerStartStopAdmin. Additional values supported
 by z/OS are RootAdmin and OperationalAdmin.' EQUALITY caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.3302 NAME ( 'ibm-pwdIndividualPolicyDN' )
 DESC 'DN of an entry containing pasword policy information. This entry can be used
 in a user entry to associate a password policy with the entry.'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.3303 NAME ( 'ibm-pwdGroupPolicyDN' ) DESC 'DN of an entry
 containing password policy information. This entry can be used in a group entry to
 associate a password policy with the entry.' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.3320 NAME ( 'ibm-slapdReplRestrictedAccess' )
 DESC 'Used to control access to the replication topology entry. If it is set to true,
 then only the root admin, local admin group members and the master DN have access to the
 replication topology entry, otherwise, any user with proper ACL may have access to the
 replication topology entry.' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.3321 NAME ( 'ibm-slapdNoReplConflictResolution' )
 DESC 'Specifies whether or not directory server will handle replication conflict
 resolution. If it is set to true, then the server does not try to compare timestamps
 for replicated entries in an attempt to resolve conflicts between the entries.
 However, conflict resolution does not apply to entry cn=schema which is always
 replaced by a replicated cn=schema.' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.3329 NAME ( 'ibm-pwdGroupAndIndividualEnabled' )
 DESC 'A value of TRUE indicates that global, group and individual password policies are
 to be considered when evaluating password policy. A value of FALSE indicates that only
 the global password policy is used.' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
```

```
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3330 NAME ( 'ibm-slapdLogMgmtStartTime' ) DESC 'specifies
 the start date and time for the log management activity. The format is YYYYMMDDHHMM'
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3331 NAME ( 'ibm-slapdLogEventFilePrefix' )
 DESC 'File name prefix for the CBE formatted log will be placed. The suffix of the CBE
 formatted log file will always be "_audit0.log" and cannot be changed. Hence if the
 prefix for the file name if specified as xyz, then the CBE formatted file name will be
 xyz_audit0.log.' EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3332 NAME ( 'ibm-slapdLogEventFormat' )
 DESC 'specifies in which event format the users want the ITDS log records' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3333 NAME ( 'ibm-slapdLogEventFilePath' ) DESC 'Log path
 for an event formatted log. On Windows, forward slashes are allowed, and a leading
 slash not preceded by a drive letter is assumed to be rooted at the install directory
 (i.e.: /tmp/ = D:\Program Files\IBM\ldap\V6.1\tmp\).' EQUALITY caseExactMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3334 NAME ( 'ibm-slapdLogMgmtFrequency' )
 DESC 'Specifies the time interval between two cycles of the log management activity'
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3335 NAME ( 'ibm-slapdAuditOperation' )
 DESC 'The audit operation for which the audit records will be converted to
 the specified event format. For example, if the attribute is set to BIND,
 then audit records related only to the bind operation will be converted to
 specified event format.' EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3342 NAME ( 'racfHavePassPhraseEnvelope' )
 DESC 'Represents the password phrase-enveloped field of the RACF user base segment'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3343 NAME ( 'racfPassPhraseEnvelope' )
 DESC 'Envelope containing user password phrase information'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3344 NAME ( 'racfKerbKeyFrom' )
 DESC 'Represents the KEYFROM field of the RACF user KERB segment'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3373 NAME ( 'ibm-slapdLogCachePath' )
 DESC 'Path where the cache files for the log management tool will be created.
 On Windows, forward slashes are allowed, and a leading slash not preceded by a
 drive letter is assumed to be rooted at the install directory (i.e.: /tmp/ =
 D:\Program Files\IBM\ldap\V6.1\tmp\).' EQUALITY caseExactMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3396 NAME ( 'passwordMaxConsecutiveRepeatedChars' )
 DESC 'Attribute used to impose the maximum number of consecutive repeated characters
 in the password field.' SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3416 NAME ( 'ibm-slapdEnableConflictResolutionForGroups' )
 DESC 'An attribute which determines whether replication conflict resolution will be
 performed from group entries or not.' EQUALITY booleanMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3417 NAME ( 'racfCfdefMaxValue' )
 DESC 'Maximum numeric value the field can contain' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3418 NAME ( 'racfUacc' )
 DESC 'Universal access authority associated with the resource profile'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3419 NAME ( 'racfStdataTrace' )
 DESC 'Whether a message is issued when this resource profile is used to assign
 an ID to the started task' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3420 NAME ( 'racfCfdefFirst' )
 DESC 'Character type restriction for the first character of the field'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3421 NAME ( 'racfStdataUser' )
 DESC 'User ID associated with this started task' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3422 NAME ( 'racfStatistics' )
 DESC 'Name of class for which RACF records statistical information'
 EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3423 NAME ( 'racfCfdefListHead' )
 DESC 'Heading for the field displayed by the LISTUSER or LISTGRP command'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3424 NAME ( 'racfControlAccessCount' )
 DESC 'Number of times that the resource profile has been referenced for control access'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3425 NAME ( 'racfGenCmd' )
 DESC 'Name of class for which RACF performs generic profile command processing'
 EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3426 NAME ( 'racfCfdefMixed' )
 DESC 'Whether mixed-case characters are allowed in the field' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3427 NAME ( 'racfGenList' ) DESC 'Name of class
 for which RACF shares in-storage generic profiles' EQUALITY caseIgnoreMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3428 NAME ( 'racfCfdefOther' )
 DESC 'Character type restriction for the characters after the first one in the field'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3429 NAME ( 'racfAccessControl' )
 DESC 'Information for controlling access to a resource, in RACF PERMIT format'
 EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3430 NAME ( 'racfApplData' )
 DESC 'Text string associated with the resource profile' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3431 NAME ( 'racfCdtinfoKeyQualifiers' )
 DESC 'Number of matching qualifiers to use when loading generic resource profile names'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3432 NAME ( 'racfCdtinfoFirst' )
 DESC 'Character type restriction for the first character of the resource profile name'
 EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3433 NAME ( 'racfCdtinfoOther' )
```

```
 DESC 'Character type restriction for the characters after the first one in a
 resource profile name' EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3434 NAME ( 'racfAutomatic' )
  DESC 'Represents the AUTOMATIC field in the RACF TAPEVOL class' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3435 NAME ( 'racfStdataTrusted' )
 DESC 'Whether this started task runs with the RACF TRUSTED attribute' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3436 NAME ( 'racfUpdateAccessCount' )
 DESC 'Number of times that the resource profile has been referenced for update access'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3437 NAME ( 'racfVolumeList' ) DESC 'Tape volume serial
 numbers represented by the resource profile' EQUALITY caseIgnoreMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3438 NAME ( 'racfTimeZone' ) DESC 'Time zone in which a terminal resides'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3439 NAME ( 'racfStdataGroup' ) DESC 'Group name associated with this started task'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3440 NAME ( 'racfStdataPrivileged' ) DESC 'Whether this started task runs
 with the RACF PRIVILEGED attribute' EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3441 NAME ( 'racfSsignonKeyEncrypted' ) DESC 'Encrypt the key value'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3442 NAME ( 'racfSsignonKeyMasked' ) DESC 'Mask the key value'
 EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3443 NAME ( 'racfSessionLock' ) DESC 'Mark the resource profile as locked'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3444 NAME ( 'racfSetroptsAttributes' ) DESC 'Additional RACF SETROPTS keywords'
 EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3445 NAME ( 'racfSigverFailLoad' ) DESC 'Conditions under which module load fails
 when digital signature verification fails' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3446 NAME ( 'racfSigverSigAudit' ) DESC 'Digital signature verification
 events to be audited' EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3447 NAME ( 'racfSigverSigRequired' ) DESC 'Whether the program object
 needs a digital signature' EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3448 NAME ( 'racfSessionSessKey' ) DESC 'Session key for this
 resource profile' EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3449 NAME ( 'racfResourceAttributes' ) DESC 'Additional resource
 profile keywords' EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3450 NAME ( 'racfResourceAudit' ) DESC 'The types of access
 to the resource profile that are logged to SMF' EQUALITY caseIgnoreMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3451 NAME ( 'racfResourceGlobalAudit' )
 DESC 'The types of access to the resource profile that are logged to SMF as set
 by a user with AUDITOR attribute' EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3452 NAME ( 'racfReadAccessCount' )
 DESC 'Number of times that the resource profile has been referenced for read access'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3453 NAME ( 'racfSessionConvSec' )
 DESC 'Level of security checking when conversations are established with the protected LU'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3454 NAME ( 'racfSessionInterval' )
 DESC 'Maximum number of days the session key is valid' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3455 NAME ( 'racfLogOptionsSuccesses' )
 DESC 'Name of class for which RACF audits successful access attempts to resources'
 EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3456 NAME ( 'racfMemberList' ) DESC 'Name of member that
 RACF is to add to the resource profile' EQUALITY caseIgnoreMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3457 NAME ( 'racfNotify' ) DESC 'User ID to notify
 whenever the resource profile is used to deny access' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3458 NAME ( 'racfLogOptionsNever' ) DESC 'Name of
 class for which RACF audits no access attempts to resources' EQUALITY caseIgnoreMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3459 NAME ( 'racfRacList' ) DESC 'Name of class for
 which RACF shares in-storage generic and discrete profiles' EQUALITY caseIgnoreMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3460 NAME ( 'racfLogOptionsFailures' )
 DESC 'Name of class for which RACF audits failed access attempts to resources'
 EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3461 NAME ( 'racfLastReferenceDate' )
 DESC 'Date when the resource profile was last referenced' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3462 NAME ( 'racfLastChangeDate' )
 DESC 'Date when the resource profile was last changed' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3463 NAME ( 'racfLevel' ) DESC 'Level number assigned
 by the installation' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3464 NAME ( 'racfLogOptionsAlways' )
 DESC 'Name of class for which RACF audits all access attempts to resources'
 EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3465 NAME ( 'racfLogOptionsDefault' )
 DESC 'Name of class for which RACF auditing is controlled by the profile
 protecting the resource' EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3466 NAME ( 'racfKerbPassword' )
 DESC 'Value of the Kerberos password for the realm' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3467 NAME ( 'racfIctxDoMap' )
 DESC 'Whether ICTX caching uses EIM mapping services' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3468 NAME ( 'racfIctxMapRequired' )
 DESC 'Whether the ICTX identity cache requires identity mapping' EQUALITY caseIgnoreMatch
```

```
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3469 NAME ( 'racfIctxUseMap' )
 DESC 'Whether the ICTX identity cache stores a valid identity mapping' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3470 NAME ( 'racfKerbDefaultTicketLife' )
 DESC 'Default ticket lifetime for the local Network Authentication Services'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3471 NAME ( 'racfKerbMinTicketLife' )
 DESC 'Minium ticket lifetime for the local Network Authentication Services'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3472 NAME ( 'racfIctxMappingTimeOut' )
 DESC 'How long the ICTX identity cache stores an identity mapping' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3473 NAME ( 'racfGlobal' )
 DESC 'Name of class for which RACF performs global access checking' EQUALITY caseIgnoreMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3474 NAME ( 'racfGeneric' ) DESC 'Name of class
 for which RACF performs generic profile checking' EQUALITY caseIgnoreMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3475 NAME ( 'racfEimX509Registry' )
 DESC 'Name of the X.509 registry in the EIM domain' EQUALITY caseExactMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3476 NAME ( 'racfEimOptions' ) DESC 'Options
 that control EIM configuration' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3477 NAME ( 'racfDlfdataJobNames' )
 DESC 'List of job names which can access the DLF objects protected by this
 resource profile' EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3478 NAME ( 'racfDlfdataRetain' )
 DESC 'Whether the DLF object can be retained after use' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3479 NAME ( 'racfEimDomainDn' )
 DESC 'Distinguished name of the EIM domain' EQUALITY caseExactMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3480 NAME ( 'racfEimKerbRegistry' )
 DESC 'Name of the Kerberos registry in the EIM domain' EQUALITY caseExactMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3481 NAME ( 'racfEimLocalRegistry' )
 DESC 'Name of the local RACF registry in the EIM domain' EQUALITY caseExactMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3482 NAME ( 'racfCopyProfileFrom' )
 DESC 'The FCLASS, FGENERIC, FROM, and FVOLUME specifications for copying the
 values from a profile' EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3483 NAME ( 'racfCfdefType' )
 DESC 'Data type of the field' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3484 NAME ( 'racfClassAct' )
 DESC 'Name of class for which RACF protection is in effect' EQUALITY caseIgnoreMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3485 NAME ( 'racfCfdefMinValue' )
 DESC 'Minimum numeric value the field can contain' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3486 NAME ( 'racfCfdefHelp' )
 DESC 'Help text for the field' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3487 NAME ( 'racfCfdefMaxLength' )
 DESC 'Maximum number of characters the field can contain' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3488 NAME ( 'racfCdtinfoSignal' )
 DESC 'Whether a signal is sent when RACLISTed resource profiles are changed'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3489 NAME ( 'racfCdtinfoRacList' )
 DESC 'Whether SETROPTS RACLIST is allowed for this class' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3490 NAME ( 'racfCdtinfoPosit' )
 DESC 'POSIT number associated with this class' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3491 NAME ( 'racfCdtinfoProfilesAllowed' )
 DESC 'Whether resource profiles can be defined for this class' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3492 NAME ( 'racfCdtinfoOperations' )
 DESC 'Whether to consider the OPERATIONS attribute when performing authorization
 checking' EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3493 NAME ( 'racfCdtinfoSecLabelsRequired' )
 DESC 'Whether a SECLABEL is required for resource profiles' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3494 NAME ( 'racfCdtinfoMacProcessing' )
 DESC 'Type of mandatory access control processing required for the class'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3495 NAME ( 'racfCdtinfoMaxLength' )
 DESC 'Maximum length of resource and resource profile names when MAXLENX is
 not specified' EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3496 NAME ( 'racfCdtinfoMaxLengthX' )
 DESC 'Maximum length of resource and resource profile names when invoking
 RACROUTE ENTITYX or when using a RACF command processor' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3497 NAME ( 'racfCdtinfoMember' )
 DESC 'Name of class grouped by the resources within this class' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3498 NAME ( 'racfCdtinfoGroup' )
 DESC 'Name of class that groups the resources within this class'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3499 NAME ( 'racfCdtinfoDefaultRc' )
 DESC 'Return code that RACF provides if a resource profile is not found'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3500 NAME ( 'racfCdtinfoCase' )
```

```
 DESC 'Whether mixed-case resource profile names are allowed for this class'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3501 NAME ( 'racfCdtinfoDefaultUacc' )
 DESC 'Minimum access allowed if the access level is not set in a resource profile'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3502 NAME ( 'racfCdtinfoGeneric' ) DESC 'Whether SETROPTS GENERIC
 and GENCMD are allowed for the class' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3503 NAME ( 'racfCdtinfoGenList' ) DESC 'Whether SETROPTS GENLIST
 is allowed for the class' EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3504 NAME ( 'racfAudit' ) DESC 'Name of class for which RACF
 performs auditing' EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3505 NAME ( 'racfAlterAccessCount' ) DESC 'Number of times
 that the resource profile has been referenced for alter access' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3506 NAME ( 'profileName' ) DESC 'Name of RACF resource profile'
 EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3508 NAME ( 'ibm-replicationWaitOnDependency' ) DESC 'Indicates
 whether the server will await the completion of the replication of dependencies prior to sending a
 replication update to a consumer.' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3509 NAME ( 'ibm-slapdReplVersion' ) DESC 'This attribute
 defines the current version of the advanced replication feature.' EQUALITY caseIgnoreMatch
 ORDERING caseIgnoreOrderingMatch SUBSTR caseIgnoreSubstringsMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.3511 NAME ( 'racfIcsfAsymUsage' ) DESC 'Allowable usage of
 an asymmetric ICSF key' EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3513 NAME ( 'racfIcsfSymExportable' ) DESC 'How symmetric
 keys covered by this profile can be exported' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3514 NAME ( 'racfIcsfSymExportCerts' ) DESC 'Digital certificate
 labels to use to export symmetric keys covered by this profile' EQUALITY caseExactMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3515 NAME ( 'racfIcsfSymExportKeys' ) DESC 'Key token labels
 for public keys to use to export symmetric keys covered by this profile' EQUALITY caseIgnoreMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3523 NAME ( 'ibm-filterBindMechanism' ) DESC 'Bind mechanism to
 use in a filter' EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3524 NAME ( 'ibm-filterConnectionEncrypted' ) DESC 'Connection
 encrypted flag to use in a filter' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3526 NAME ( 'ibm-filterDayOfWeek' ) DESC 'Directory entry
 access day of week to use in a filter.  The value is an integer mapping the days of the week
 as follows: Sunday = 0, Monday = 1, Tuesday = 2, Wednesday = 3, Thursday = 4, Friday = 5,
 Saturday = 6.' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3527 NAME ( 'ibm-filterIP' ) DESC 'IPv4 or IPv6 IP address of
 a client connection to use in a filter' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3528 NAME ( 'ibm-filterSubject' ) DESC 'Distinguished name to
 use in a filter' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3529 NAME ( 'ibm-filterTimeOfDay' ) DESC 'Directory entry
 access time of day to use in a filter.  The value is the hh:mm format of 24 hour time, with hh
 ranging from 00 to 23 and mm ranging from 00 to 59.' EQUALITY caseIgnoreMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3559 NAME ( 'ibm-slapdSAFSecurityDomain' ) DESC 'The high level
 component of the profile names used to define LDAP-related information in the z/OS Security Manager.'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3560 NAME ( 'racfKerbCheckAddrs' ) DESC 'Whether the Kerberos server
 is to take addresses in the tickets into account when it performs ticket operations'
 EQUALITY caseIgnoreMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3656 NAME ( 'ibm-slapdServerCompatibilityLevel' ) DESC 'The server
 compatibility level used to determine functional capability.'
 EQUALITY integerMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.27  SINGLE-VALUE USAGE dSAOperation
attributetypes=( 1.3.18.0.2.4.454 NAME ( 'passwordMaxRepeatedChars' ) DESC '  '
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.469 NAME ( 'passwordMinOtherChars' ) DESC '  '
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.470 NAME ( 'ibmAttributeTypes' ) DESC 'IBM attribute types'
 SYNTAX 1.3.18.0.2.8.1 USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.473 NAME ( 'passwordMinAlphaChars' ) DESC 'Specifies the minimum
 number of characters required for a users password.' SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.499 NAME ( 'passwordMinDiffChars' ) DESC 'Specifies the minimum
 number of different (unique) characters required for a users password.'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 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'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.827 NAME ( 'racfOmvsMaximumCPUTime' ) DESC 'Represents the
 CPUTIMEMAX(cpu-time) field of the RACF OMVS SEGMENT' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.828 NAME ( 'racfOmvsMaximumFilesPerProcess' ) DESC 'Represents the
 FILEPROCMAX(files-per-process) field of the RACF OMVS SEGMENT' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.829 NAME ( 'racfOmvsMaximumMemoryMapArea' ) DESC 'Represents the
 MMAPAREAMAX(memory-map-size) field of the RACF OMVS SEGMENT' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.830 NAME ( 'racfOmvsMaximumProcessesPerUID' ) DESC 'Represents the
 PROCUSERMAX(processes-per-UID) field of the RACF OMVS SEGMENT' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.831 NAME ( 'racfOmvsMaximumThreadsPerProcess' ) DESC 'Represents the
 THREADSMAX(threads-per-process) field of the RACF OMVS SEGMENT' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
 USAGE userApplications )
attributetypes=( 1.3.6.1.1.4 NAME ( 'vendorName' ) DESC 'Name of the company that implemented the LDAP server'
 EQUALITY caseExactMatch SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE dSAOperation )
attributetypes=( 1.3.6.1.1.5 NAME ( 'vendorVersion' ) DESC 'Version of the LDAP Server implementation'
 EQUALITY caseExactMatch SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE dSAOperation )
attributetypes=( 1.3.6.1.4.1.1466.101.120.13 NAME ( 'supportedControl' ) DESC 'Controls supported by this server'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 USAGE dSAOperation )
attributetypes=( 1.3.6.1.4.1.1466.101.120.14 NAME ( 'supportedSASLMechanisms' ) DESC 'SASL mechanisms
 supported by this server' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE dSAOperation )
attributetypes=( 1.3.6.1.4.1.1466.101.120.15 NAME ( 'supportedLDAPVersion' ) DESC 'LDAP protocol versions
 supported by this server' SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE dSAOperation )
attributetypes=( 1.3.6.1.4.1.1466.101.120.16 NAME ( 'ldapSyntaxes' ) DESC 'LDAP syntaxes'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.54 USAGE directoryOperation )
```

```
attributetypes=( 1.3.6.1.4.1.1466.101.120.5 NAME ( 'namingContexts' ) DESC 'LDAP server naming contexts'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE dSAOperation )
attributetypes=( 1.3.6.1.4.1.1466.101.120.6 NAME ( 'altServer' ) DESC 'Alternate LDAP server'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE dSAOperation )
attributetypes=( 1.3.6.1.4.1.1466.101.120.7 NAME ( 'supportedExtension' ) DESC 'Extensions supported
 by this server' SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 USAGE dSAOperation )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.1 NAME ( 'pwdAttribute' ) DESC 'Specifies the name of the
 attribute to which the password policy is applied, ie  userPassword' EQUALITY objectIdentifierMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 USAGE userApplications )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.10 NAME ( 'pwdLockoutDuration' ) DESC 'Specifies the number of
 seconds that the password cannot be used to authenticate due to too many failed bind attempts.'
 EQUALITY integerMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.11 NAME ( 'pwdMaxFailure' ) DESC 'Specifies the number of
 consecutive failed bind attempts after which the password may not be used to authenticate.'
 EQUALITY integerMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.12 NAME ( 'pwdFailureCountInterval' ) DESC 'Specifies the
 number of seconds after which the password failures are purged from the failure counter, even though
 no successful authentication occurred.' EQUALITY integerMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 USAGE userApplications )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.13 NAME ( 'pwdMustChange' ) DESC 'Specifies with a value of
 TRUE that users must change their passwords when they first bind to the directory after a password
 is set or reset by the administrator.' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE userApplications )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.14 NAME ( 'pwdAllowUserChange' ) DESC 'Indicates whether users
 can change their own passwords.' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE userApplications )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.15 NAME ( 'pwdSafeModify' ) DESC 'Specifies whether or not the
 existing password must be sent when changing a password' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
 USAGE userApplications )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.16 NAME ( 'pwdChangedTime' ) DESC 'Specifies the last time the
 entrys password was changed' EQUALITY generalizedTimeMatch ORDERING generalizedTimeOrderingMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 USAGE directoryOperation )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.17 NAME ( 'pwdAccountLockedTime' ) DESC 'Specifies the time
 that the users account was locked' EQUALITY generalizedTimeMatch ORDERING generalizedTimeOrderingMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 USAGE directoryOperation )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.18 NAME ( 'pwdExpirationWarned' ) DESC 'The time the user was
 first warned about the coming  expiration of the password' EQUALITY generalizedTimeMatch ORDERING
 generalizedTimeOrderingMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 USAGE directoryOperation )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.19 NAME ( 'pwdFailureTime' ) DESC 'The timestamps of the last
 consecutive authentication failures' EQUALITY generalizedTimeMatch ORDERING generalizedTimeOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 USAGE directoryOperation )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.2 NAME ( 'pwdMinAge' ) DESC 'Specifies in seconds, the period of
 time a password must be in effect before a user can change it.' EQUALITY integerMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.20 NAME ( 'pwdHistory' ) DESC 'The history of users passwords'
 EQUALITY caseExactMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE directoryOperation )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.21 NAME ( 'pwdGraceUseTime' ) DESC 'The timestamps of the
 grace login once the password has expired' EQUALITY generalizedTimeMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
 USAGE directoryOperation )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.22 NAME ( 'pwdReset' ) DESC 'Indicates that the password has been reset.'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE directoryOperation )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.3 NAME ( 'pwdMaxAge' ) DESC 'Specifies in seconds, the period of
 time password can be used before they expire.' EQUALITY integerMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.4 NAME ( 'pwdInHistory' ) DESC 'Specifies the number of passwords
 which are stored in the pwdHistory attribute.' EQUALITY integerMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.5 NAME ( 'pwdCheckSyntax' ) DESC 'Indicates how the password
 syntax will be checked while being modified or added' EQUALITY integerMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.6 NAME ( 'pwdMinLength' ) DESC 'Holds the minimum number of
 characters that must be used in a password' EQUALITY integerMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.7 NAME ( 'pwdExpireWarning' ) DESC 'Specifies the maximum
 number of seconds before a password is due to expire that expiration warning messages will be returned
 to an authenticating user.' EQUALITY integerMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 USAGE userApplications )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.8 NAME ( 'pwdGraceLoginLimit' ) DESC 'Specifies the number
 of times an expired password can be used to authenticate' EQUALITY integerMatch
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 1.3.6.1.4.1.42.2.27.8.1.9 NAME ( 'pwdLockout' ) DESC 'Indicates, when its value is TRUE,
 that the password may not be used to authenticate after a specified number of consecutive failed bind
 attempts' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE userApplications )
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 entry attribute' SINGLE-VALUE NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 USAGE userApplications )
attributetypes=( 2.16.840.1.113730.3.1.11 NAME ( 'newSuperior' ) DESC 'Specifies the name of the new
 superior of the existing entry' EQUALITY distinguishedNameMatch SINGLE-VALUE NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userApplications )
attributetypes=( 2.16.840.1.113730.3.1.198 NAME ( 'memberURL' ) DESC 'Specifies a URL associated with
 each member of a group' EQUALITY caseExactMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 2.16.840.1.113730.3.1.34 NAME ( 'ref' ) DESC 'Specifies the URI to continue the
 LDAP operation' EQUALITY caseExactMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 2.16.840.1.113730.3.1.35 NAME ( 'changeLog' ) DESC 'Distinguished name of the server
 change log' EQUALITY distinguishedNameMatch NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
 USAGE dSAOperation )
attributetypes=( 2.16.840.1.113730.3.1.5 NAME ( 'changeNumber' ) DESC 'Contains the assigned change
 number for the modification' EQUALITY integerMatch SINGLE-VALUE NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 USAGE userApplications )
attributetypes=( 2.16.840.1.113730.3.1.6 NAME ( 'targetDN' ) DESC 'Defines the distinguished name of
 an entry that was modified' EQUALITY distinguishedNameMatch SINGLE-VALUE NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userApplications )
attributetypes=( 2.16.840.1.113730.3.1.7 NAME ( 'changeType' ) DESC 'Describes the type of change
 performed on an entry (add, modify, delete, modrdn)' EQUALITY caseIgnoreMatch SINGLE-VALUE NO-USER-MODIFICATION
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 2.16.840.1.113730.3.1.77 NAME ( 'changeTime' ) DESC 'Time last changed'
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 USAGE userApplications )
attributetypes=( 2.16.840.1.113730.3.1.8 NAME ( 'changes' ) DESC 'Defines changes made to a directory
 server (LDIF format)' SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 USAGE userApplications )
attributetypes=( 2.16.840.1.113730.3.1.9 NAME ( 'newRdn' ) DESC 'The new RDN of an entry'
 EQUALITY distinguishedNameMatch SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
 USAGE userApplications )
attributetypes=( 2.5.18.1 NAME ( 'createTimestamp' ) DESC 'Entry creation time'
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 USAGE directoryOperation )
attributetypes=( 2.5.18.10 NAME ( 'subschemaSubentry' ) DESC 'Schema associated with an entry'
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE directoryOperation )
attributetypes=( 2.5.18.2 NAME ( 'modifyTimestamp' ) DESC 'Time of last entry modification'
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 USAGE directoryOperation )
```

```
attributetypes=( 2.5.18.3 NAME ( 'creatorsName' ) DESC 'Name of entry creator'
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE directoryOperation )
attributetypes=( 2.5.18.4 NAME ( 'modifiersName' ) DESC 'Name of last entry modifier'
 SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE directoryOperation )
attributetypes=( 2.5.18.6 NAME ( 'subtreeSpecification' ) DESC 'Subtree specification'
 SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE directoryOperation )
attributetypes=( 2.5.18.9 NAME ( 'hasSubordinates' ) DESC 'Indicates whether any subordinate entries
 exist below the entry holding this attribute.' SINGLE-VALUE NO-USER-MODIFICATION SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
 USAGE directoryOperation )
attributetypes=( 2.5.21.1 NAME ( 'ditStructureRules' ) DESC 'Directory structure rules'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.17 USAGE directoryOperation )
attributetypes=( 2.5.21.2 NAME ( 'ditContentRules' ) DESC 'Directory content rules'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.16 USAGE directoryOperation )
attributetypes=( 2.5.21.4 NAME ( 'matchingRules' ) DESC 'LDAP matching rules'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.30 USAGE directoryOperation )
attributetypes=( 2.5.21.5 NAME ( 'attributeTypes' ) DESC 'LDAP attribute types'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.3 USAGE directoryOperation )
attributetypes=( 2.5.21.6 NAME ( 'objectClasses' ) DESC 'LDAP object classes'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.37 USAGE directoryOperation )
attributetypes=( 2.5.21.7 NAME ( 'nameForms' ) DESC 'Directory name forms'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.35 USAGE directoryOperation )
attributetypes=( 2.5.21.8 NAME ( 'matchingRuleUse' ) DESC 'LDAP matching rule uses'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.31 USAGE directoryOperation )
attributetypes=( 2.5.4.0 NAME ( 'objectClass' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.38
 USAGE userApplications )
attributetypes=( 2.5.4.1 NAME ( 'aliasedObjectName' 'aliasedEntryName' ) DESC 'True name for an
 alias entry' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE userApplications )
attributetypes=( 2.5.4.10 NAME ( 'o' 'organizationName' 'organization' ) DESC 'The name of an
 organization' SUP name USAGE userApplications )
attributetypes=( 2.5.4.11 NAME ( 'ou' 'organizationalUnit' 'organizationalUnitName' ) DESC 'The name
 of an organizational unit' SUP name USAGE userApplications )
attributetypes=( 2.5.4.13 NAME ( 'description' ) DESC 'Provides a description of a directory entry'
 EQUALITY caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 2.5.4.15 NAME ( 'businessCategory' ) DESC 'Describes the kind of business performed
 by an organization' EQUALITY caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 2.5.4.3 NAME ( 'cn' 'commonName' ) SUP name USAGE userApplications )
attributetypes=( 2.5.4.31 NAME ( 'member' ) DESC 'Defines a member of a set' SUP dn USAGE userApplications )
attributetypes=( 2.5.4.32 NAME ( 'owner' ) DESC 'Specifies the DN of the person responsible
 for the entry' SUP dn USAGE userApplications )
attributetypes=( 2.5.4.34 NAME ( 'seeAlso' ) DESC 'Identifies another entry that may contain
 information related this entry' SUP dn USAGE userApplications )
attributetypes=( 2.5.4.35 NAME ( 'userPassword' ) DESC 'Defines the user password'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 USAGE userApplications )
attributetypes=( 2.5.4.41 NAME ( 'name' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
attributetypes=( 2.5.4.49 NAME ( 'dn' 'distinguishedName' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
 USAGE userApplications )
attributetypes=( 2.5.4.50 NAME ( 'uniqueMember' ) DESC 'Defines a member of a set' SUP dn
 USAGE userApplications )
attributetypes=( 2.5.4.6 NAME ( 'c' 'countryName' ) DESC 'A two-letter ISO 3166 country code' SUP name
 SINGLE-VALUE USAGE userApplications )
attributetypes=( 2.5.4.7 NAME ( 'l' 'localityName' ) DESC 'The name of a locality, such as a city,
 county or other geographic region' SUP name USAGE userApplications )
attributetypes=( 2.5.4.8 NAME ( 'st' 'stateOrProvince' 'stateOrProvinceName' )
 DESC 'The full name of a state or province' SUP name USAGE userApplications )
ibmattributetypes=( 0.9.2342.19200300.100.1.1 ACCESS-CLASS normal )
ibmattributetypes=( 0.9.2342.19200300.100.1.23 ACCESS-CLASS system )
ibmattributetypes=( 0.9.2342.19200300.100.1.24 ACCESS-CLASS system )
ibmattributetypes=( 1.2.840.113556.1.4.656 ACCESS-CLASS normal )
ibmattributetypes=( 1.2.840.113556.1.4.77 ACCESS-CLASS normal )
ibmattributetypes=( 1.2.840.113556.1.4.867 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.1068 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.1088 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.1091 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=( 1.3.18.0.2.4.1144 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=( 1.3.18.0.2.4.1152 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.1153 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.1154 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.1155 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.1156 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.1157 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.1158 ACCESS-CLASS critical )
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=( 1.3.18.0.2.4.155 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.1780 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.185 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.186 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.187 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=( 1.3.18.0.2.4.1913 ACCESS-CLASS sensitive )
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=( 1.3.18.0.2.4.195 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=( 1.3.18.0.2.4.199 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.200 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.2007 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=( 1.3.18.0.2.4.204 ACCESS-CLASS critical )
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=( 1.3.18.0.2.4.214 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=( 1.3.18.0.2.4.221 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.222 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.223 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2239 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.224 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=( 1.3.18.0.2.4.2242 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2243 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2244 ACCESS-CLASS system )
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=( 1.3.18.0.2.4.2325 DBNAME( 'entryChecksum'  'entryChecksum' )
 ACCESS-CLASS system LENGTH 100 )
ibmattributetypes=( 1.3.18.0.2.4.2326 DBNAME( 'entryChecksumOp'  'entryChecksumOp' )
 ACCESS-CLASS system LENGTH 100 )
ibmattributetypes=( 1.3.18.0.2.4.2327 DBNAME( 'supportedReplicat'  'supportedReplicat' )
 ACCESS-CLASS system LENGTH 240 )
ibmattributetypes=( 1.3.18.0.2.4.2328 DBNAME( 'serverId'  'serverId' )
 ACCESS-CLASS system LENGTH 240 )
ibmattributetypes=( 1.3.18.0.2.4.2329 DBNAME( 'replicaType'  'replicaType' )
 ACCESS-CLASS normal LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.233 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2330 DBNAME( 'replicationChange'  'replicationChange' )
 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2331 DBNAME( 'effectiveReplicat'  'effectiveReplicat' )
 ACCESS-CLASS system LENGTH 240 )
ibmattributetypes=( 1.3.18.0.2.4.2332 DBNAME( 'replicationLastAd'  'replicationLastAd' )
 ACCESS-CLASS system LENGTH 1024 )
ibmattributetypes=( 1.3.18.0.2.4.2333 DBNAME( 'replicationPendin'  'replicationPendin' )
 ACCESS-CLASS system LENGTH 12 )
ibmattributetypes=( 1.3.18.0.2.4.2334 DBNAME( 'replicationLastCh'  'replicationLastCh' )
 ACCESS-CLASS system LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.2335 DBNAME( 'replicationLastFi'  'replicationLastFi' )
 ACCESS-CLASS system LENGTH 30 )
ibmattributetypes=( 1.3.18.0.2.4.2336 DBNAME( 'replicationState'  'replicationState' )
 ACCESS-CLASS system LENGTH 240 )
ibmattributetypes=( 1.3.18.0.2.4.2337 DBNAME( 'replicationPendch'  'replicationPendch' )
 ACCESS-CLASS system LENGTH 1100 )
ibmattributetypes=( 1.3.18.0.2.4.2338 DBNAME( 'replicationLastAc'  'replicationLastAc' )
 ACCESS-CLASS system LENGTH 32 )
ibmattributetypes=( 1.3.18.0.2.4.2339 DBNAME( 'replicationNextTi'  'replicationNextTi' )
 ACCESS-CLASS system LENGTH 30 )
ibmattributetypes=( 1.3.18.0.2.4.234 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2340 DBNAME( 'replicationLastRe'  'replicationLastRe' )
 ACCESS-CLASS system LENGTH 2048 )
ibmattributetypes=( 1.3.18.0.2.4.2341 DBNAME( 'replicationBatchS'  'replicationBatchS' )
 ACCESS-CLASS normal LENGTH 7 )
ibmattributetypes=( 1.3.18.0.2.4.2342 DBNAME( 'replicaKeyFile'  'replicaKeyFile' )
 ACCESS-CLASS normal LENGTH 1000 )
ibmattributetypes=( 1.3.18.0.2.4.2343 DBNAME( 'replicaKeylabel'  'replicaKeylabel' )
 ACCESS-CLASS normal LENGTH 240 )
ibmattributetypes=( 1.3.18.0.2.4.2344 DBNAME( 'scheduleTuesday'  'scheduleTuesday' )
 ACCESS-CLASS normal LENGTH 1000 )
ibmattributetypes=( 1.3.18.0.2.4.2345 DBNAME( 'replicationTimesU'  'replicationTimesU' )
 ACCESS-CLASS normal LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.2346 DBNAME( 'scheduleFriday'  'scheduleFriday' )
 ACCESS-CLASS normal LENGTH 1000 )
ibmattributetypes=( 1.3.18.0.2.4.2347 DBNAME( 'scheduleSaturday'  'scheduleSaturday' )
 ACCESS-CLASS normal LENGTH 1000 )
ibmattributetypes=( 1.3.18.0.2.4.2348 DBNAME( 'scheduleWeds'  'scheduleWeds' )
 ACCESS-CLASS normal LENGTH 1000 )
ibmattributetypes=( 1.3.18.0.2.4.2349 DBNAME( 'replicationOnHold'  'replicationOnHold' )
 ACCESS-CLASS normal LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.235 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2350 DBNAME( 'scheduleSunday'  'scheduleSunday' )
 ACCESS-CLASS normal LENGTH 1000 )
ibmattributetypes=( 1.3.18.0.2.4.2351 DBNAME( 'replicaCredsDN'  'replicaCredsDN' )
 ACCESS-CLASS normal LENGTH 1000 EQUALITY )
ibmattributetypes=( 1.3.18.0.2.4.2352 DBNAME( 'replicationImmed'  'replicationImmed' )
 ACCESS-CLASS normal LENGTH 7 )
ibmattributetypes=( 1.3.18.0.2.4.2353 DBNAME( 'scheduleMonday'  'scheduleMonday' )
 ACCESS-CLASS normal LENGTH 1000 )
ibmattributetypes=( 1.3.18.0.2.4.2354 DBNAME( 'replicaKeypwd'  'replicaKeypwd' )
 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.2355 DBNAME( 'replicaScheduleDN'  'replicaScheduleDN' )
 ACCESS-CLASS normal LENGTH 1000 EQUALITY )
ibmattributetypes=( 1.3.18.0.2.4.2356 DBNAME( 'scheduleThursday'  'scheduleThursday' )
 ACCESS-CLASS normal LENGTH 1000 )
ibmattributetypes=( 1.3.18.0.2.4.2357 DBNAME( 'replicaConsumerID'  'replicaConsumerID' )
 ACCESS-CLASS normal LENGTH 128 )
ibmattributetypes=( 1.3.18.0.2.4.2358 DBNAME( 'replicaReferralUR'  'replicaReferralUR' )
 ACCESS-CLASS normal LENGTH 2048 )
```

```
ibmattributetypes=( 1.3.18.0.2.4.2359 DBNAME( 'replicaID'  'replicaID' )
 ACCESS-CLASS normal LENGTH 240 )
ibmattributetypes=( 1.3.18.0.2.4.236 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2360 DBNAME( 'replicaURL'  'replicaURL' )
 ACCESS-CLASS normal LENGTH 128 )
ibmattributetypes=( 1.3.18.0.2.4.2361 DBNAME( 'replicaGroup'  'replicaGroup' )
 ACCESS-CLASS normal LENGTH 240 )
ibmattributetypes=( 1.3.18.0.2.4.2367 DBNAME( 'slapdReplicaSubtr'  'slapdReplicaSubtr' )
 ACCESS-CLASS normal LENGTH 1000 )
ibmattributetypes=( 1.3.18.0.2.4.237 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2376 DBNAME( 'slapdPageNonAdmn'  'slapdPageNonAdmn' )
 ACCESS-CLASS critical LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.2377 DBNAME( 'slapdSortNonAdmin'  'slapdSortNonAdmin' )
 ACCESS-CLASS critical LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.238 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2380 DBNAME( 'slapdPageResLmt'  'slapdPageResLmt' )
 ACCESS-CLASS critical LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.2381 DBNAME( 'slapdSortKeyLimit'  'slapdSortKeyLimit' )
 ACCESS-CLASS critical LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.239 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.240 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2401 DBNAME( 'MasterReferral'  'MasterReferral' )
 ACCESS-CLASS critical LENGTH 256 )
ibmattributetypes=( 1.3.18.0.2.4.2409 DBNAME( 'MasterDN'  'MasterDN' )
 ACCESS-CLASS critical LENGTH 1000 )
ibmattributetypes=( 1.3.18.0.2.4.241 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2411 DBNAME( 'MasterPW'  'MasterPW' )
 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.242 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2420 DBNAME ( 'slapdKrbAdminDN'  'slapdKrbAdminDN' )
 ACCESS-CLASS critical LENGTH 512 )
ibmattributetypes=( 1.3.18.0.2.4.2425 DBNAME ( 'slapdAdminPW'  'slapdAdminPW' )
 ACCESS-CLASS critical LENGTH 128 )
ibmattributetypes=( 1.3.18.0.2.4.2428 DBNAME ( 'slapdAdminDN'  'slapdAdminDN' )
 ACCESS-CLASS critical LENGTH 1000 )
ibmattributetypes=( 1.3.18.0.2.4.243 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2430 DBNAME( 'slapdInvalidLine'  'slapdInvalidLine' )
 ACCESS-CLASS normal LENGTH 1024 )
ibmattributetypes=( 1.3.18.0.2.4.2433 DBNAME( 'slapdServerId'  'slapdServerId' )
 ACCESS-CLASS normal LENGTH 240 )
ibmattributetypes=( 1.3.18.0.2.4.244 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2449 ACCESS-CLASS system )
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=( 1.3.18.0.2.4.2481 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2482 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2484 DBNAME( 'ibmreplexclcap'  'ibmreplexclcap' )
 ACCESS-CLASS normal LENGTH 100 )
ibmattributetypes=( 1.3.18.0.2.4.2486 DBNAME( 'slapdMaxPendingCh'  'slapdMaxPendingCh' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.249 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2493 DBNAME ( 'pwdPolicy' 'pwdPolicy' )
 ACCESS-CLASS normal LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.2494 DBNAME( 'replDailySchedNam'  'replDailySchedNam' )
 ACCESS-CLASS normal LENGTH 200 )
ibmattributetypes=( 1.3.18.0.2.4.2495 DBNAME( 'replThisSvrMast'  'replThisSvrMast' )
 ACCESS-CLASS system LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.2496 DBNAME( 'replCredName'  'replCredName' )
 ACCESS-CLASS normal LENGTH 200 )
ibmattributetypes=( 1.3.18.0.2.4.2497 DBNAME( 'replWeeklySchedNa'  'replWeeklySchedNa' )
 ACCESS-CLASS normal LENGTH 200 )
ibmattributetypes=( 1.3.18.0.2.4.2498 DBNAME( 'replIsQuiesced'  'replIsQuiesced' )
 ACCESS-CLASS system LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.250 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2500 DBNAME( 'slapdMigrationInf'  'slapdMigrationInf' )
 ACCESS-CLASS critical LENGTH 2048 )
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=( 1.3.18.0.2.4.256 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=( 1.3.18.0.2.4.260 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=( 1.3.18.0.2.4.265 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=( 1.3.18.0.2.4.268 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=( 1.3.18.0.2.4.275 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.276 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=( 1.3.18.0.2.4.279 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=( 1.3.18.0.2.4.285 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=( 1.3.18.0.2.4.288 ACCESS-CLASS restricted )
ibmattributetypes=( 1.3.18.0.2.4.289 ACCESS-CLASS restricted )
```

```
ibmattributetypes=( 1.3.18.0.2.4.290 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.298 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.299 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.300 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.301 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3013 DBNAME ( 'AdmGroupEnabled'  'AdmGroupEnabled' )
 ACCESS-CLASS critical LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.302 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.303 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3032 DBNAME ( 'DigestAdminUser'  'DigestAdminUser' )
 ACCESS-CLASS critical LENGTH 512 )
ibmattributetypes=( 1.3.18.0.2.4.304 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3072 DBNAME( 'SearchSizeLimit'  'SearchSizeLimit' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.3073 DBNAME( 'SearchTimeLimit'  'SearchTimeLimit' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.3081 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3089 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3090 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3091 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3094 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3095 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3097 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3098 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3128 DBNAME( 'ibmlog'  'ibmlog' )
 ACCESS-CLASS critical LENGTH 1024 )
ibmattributetypes=( 1.3.18.0.2.4.3129 DBNAME( 'logMaxArchives'  'logMaxArchives' )
 ACCESS-CLASS critical LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.3130 DBNAME( 'logOptions'  'logOptions' )
 ACCESS-CLASS critical LENGTH 30 )
ibmattributetypes=( 1.3.18.0.2.4.3131 DBNAME( 'logSizeThreshold'  'logSizeThreshold' )
 ACCESS-CLASS critical LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.3134 DBNAME( 'logArchivePath'  'logArchivePath' )
 ACCESS-CLASS critical LENGTH 1024 )
ibmattributetypes=( 1.3.18.0.2.4.3138 DBNAME( 'replFailures'  'replFailures' )
 ACCESS-CLASS normal LENGTH 10 )
ibmattributetypes=( 1.3.18.0.2.4.3139 DBNAME( 'replicationFailed'  'replicationFailed' )
 ACCESS-CLASS normal LENGTH 1100 )
ibmattributetypes=( 1.3.18.0.2.4.3141 DBNAME ( 'pwdAccountLocked' 'pwdAccountLocked' )
 ACCESS-CLASS critical LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.3142 DBNAME( 'slapdReplConflict'  'slapdReplConflict' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.3150 DBNAME( 'replmeth'  'replicaMeth' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.3151 DBNAME( 'replicacon'  'replicaCon' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.3152 DBNAME( 'slapdReplMaxError'  'slapdReplMaxError' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.3153 DBNAME( 'slapdReplContextC'  'slapdReplContextC' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.3215 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3216 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3223 DBNAME( 'replperf'  'replperf' )
 ACCESS-CLASS normal LENGTH 500 )
ibmattributetypes=( 1.3.18.0.2.4.3238 DBNAME ( 'pwdPolicyStartTim' 'pwdPolicyStartTim' )
 ACCESS-CLASS normal LENGTH 30 )
ibmattributetypes=( 1.3.18.0.2.4.3239 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3240 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3241 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3242 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3243 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3244 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3245 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3261 DBNAME( 'logCARSOptions'  'logCARSOptions' )
 ACCESS-CLASS critical LENGTH 1024 )
ibmattributetypes=( 1.3.18.0.2.4.3263 DBNAME( 'logCARSPort'  'logCARSPort' )
 ACCESS-CLASS critical LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.3264 DBNAME( 'eventFileOptions'  'eventFileOptions' )
 ACCESS-CLASS critical LENGTH 1024 )
ibmattributetypes=( 1.3.18.0.2.4.3265 DBNAME( 'logCARSServer'  'logCARSServer' )
 ACCESS-CLASS critical LENGTH 128 )
ibmattributetypes=( 1.3.18.0.2.4.3266 DBNAME( 'eventFileSizThres'  'eventFileSizThres' )
 ACCESS-CLASS critical LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.3267 DBNAME( 'eventMaxArchives'  'eventMaxArchives' )
 ACCESS-CLASS critical LENGTH 1024 )
ibmattributetypes=( 1.3.18.0.2.4.3268 DBNAME( 'eventArchivePath'  'eventArchivePath' )
 ACCESS-CLASS critical LENGTH 1024 )
ibmattributetypes=( 1.3.18.0.2.4.3269 DBNAME( 'logCARSEnabled'  'logCARSEnabled' )
 ACCESS-CLASS critical LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.3270 DBNAME( 'eventFileEnabled'  'eventFileEnabled' )
 ACCESS-CLASS critical LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.3288 DBNAME( 'replicaPKCS11'  'replicaPKCS11' )
 ACCESS-CLASS normal LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.3294 DBNAME( 'replCreatMissEntr'  'replCreatMissEntr' )
 ACCESS-CLASS normal LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.3295 DBNAME( 'replFilterDN'  'replFilterDN' )
 ACCESS-CLASS normal LENGTH 1000 )
ibmattributetypes=( 1.3.18.0.2.4.3296 DBNAME( 'replicationFilter'  'replicationFilter' )
 ACCESS-CLASS normal LENGTH 1024 EQUALITY )
ibmattributetypes=( 1.3.18.0.2.4.3299 DBNAME ( 'slapdAdminRole'  'slapdAdminRole' )
 ACCESS-CLASS critical LENGTH 100 )
ibmattributetypes=( 1.3.18.0.2.4.3302 DBNAME ( 'pwdIndividualPol' 'pwdIndividualPol' )
 ACCESS-CLASS critical LENGTH 1000 )
ibmattributetypes=( 1.3.18.0.2.4.3303 DBNAME ( 'pwdGroupPolicyDN' 'pwdGroupPolicyDN' )
 ACCESS-CLASS critical LENGTH 1000 )
ibmattributetypes=( 1.3.18.0.2.4.3320 DBNAME( 'ReplAccess'  'ReplAccess' )
 ACCESS-CLASS critical LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.3321 DBNAME( 'NoReplConflict'  'NoReplConflict' )
 ACCESS-CLASS critical LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.3329 DBNAME ( 'pwdGroupAndIndiv' 'pwdGroupAndIndiv' )
 ACCESS-CLASS normal LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.3330 DBNAME( 'logMgmtStartTime'  'logMgmtStartTime' )
 ACCESS-CLASS critical LENGTH 12 )
ibmattributetypes=( 1.3.18.0.2.4.3331 DBNAME( 'eventFilePrefix'  'eventFilePrefix' )
 ACCESS-CLASS critical LENGTH 1017 )
ibmattributetypes=( 1.3.18.0.2.4.3332 DBNAME( 'logEventFormat'  'logEventFormat' )
```

```
 ACCESS-CLASS critical LENGTH 30 )
ibmattributetypes=( 1.3.18.0.2.4.3333 DBNAME( 'logEventFilePath'  'logEventFilePath' )
 ACCESS-CLASS critical LENGTH 1024 )
ibmattributetypes=( 1.3.18.0.2.4.3334 DBNAME( 'logMgmtFrequency'  'logMgmtFrequency' )
 ACCESS-CLASS critical LENGTH 10 )
ibmattributetypes=( 1.3.18.0.2.4.3335 DBNAME( 'auditOperation'  'auditOperation' )
 ACCESS-CLASS critical LENGTH 30 )
ibmattributetypes=( 1.3.18.0.2.4.3342 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3343 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3344 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3373 DBNAME( 'logCachePath'  'logCachePath' )
 ACCESS-CLASS critical LENGTH 1024 )
ibmattributetypes=( 1.3.18.0.2.4.3396 DBNAME ( 'pwMaxConRepChars' 'pwMaxConRepChars' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.3416 DBNAME( 'slapdEnableConfli'  'slapdEnableConfli' )
 ACCESS-CLASS normal LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.3417 DBNAME( 'racfCfdefMaxValue'  'racfCfdefMaxValue' )
 ACCESS-CLASS sensitive LENGTH 15 )
ibmattributetypes=( 1.3.18.0.2.4.3418 DBNAME( 'racfUacc'  'racfUacc' )
 ACCESS-CLASS sensitive LENGTH 15 )
ibmattributetypes=( 1.3.18.0.2.4.3419 DBNAME( 'racfStdataTrace'  'racfStdataTrace' )
 ACCESS-CLASS sensitive LENGTH 3 )
ibmattributetypes=( 1.3.18.0.2.4.3420 DBNAME( 'racfCfdefFirst'  'racfCfdefFirst' )
 ACCESS-CLASS sensitive LENGTH 15 )
ibmattributetypes=( 1.3.18.0.2.4.3421 DBNAME( 'racfStdataUser'  'racfStdataUser' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3422 DBNAME( 'racfStatistics'  'racfStatistics' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3423 DBNAME( 'racfCfdefListHead'  'racfCfdefListHead' )
 ACCESS-CLASS sensitive LENGTH 40 )
ibmattributetypes=( 1.3.18.0.2.4.3424 DBNAME( 'racfControlAccess'  'racfControlAccess' )
 ACCESS-CLASS sensitive LENGTH 15 )
ibmattributetypes=( 1.3.18.0.2.4.3425 DBNAME( 'racfGenCmd'  'racfGenCmd' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3426 DBNAME( 'racfCfdefMixed'  'racfCfdefMixed' )
 ACCESS-CLASS sensitive LENGTH 3 )
ibmattributetypes=( 1.3.18.0.2.4.3427 DBNAME( 'racfGenList'  'racfGenList' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3428 DBNAME( 'racfCfdefOther'  'racfCfdefOther' )
 ACCESS-CLASS sensitive LENGTH 15 )
ibmattributetypes=( 1.3.18.0.2.4.3429 DBNAME( 'racfAccessControl'  'racfAccessControl' )
 ACCESS-CLASS sensitive  LENGTH 4096 )
ibmattributetypes=( 1.3.18.0.2.4.3430 DBNAME( 'racfApplData'  'racfApplData' )
 ACCESS-CLASS sensitive LENGTH 255 )
ibmattributetypes=( 1.3.18.0.2.4.3431 DBNAME( 'racfCdtinfoKeyQua'  'racfCdtinfoKeyQua' )
 ACCESS-CLASS sensitive LENGTH 3 )
ibmattributetypes=( 1.3.18.0.2.4.3432 DBNAME( 'racfCdtinfoFirst'  'racfCdtinfoFirst' )
 ACCESS-CLASS sensitive LENGTH 10 )
ibmattributetypes=( 1.3.18.0.2.4.3433 DBNAME( 'racfCdtinfoOther'  'racfCdtinfoOther' )
 ACCESS-CLASS sensitive LENGTH 10 )
ibmattributetypes=( 1.3.18.0.2.4.3434 DBNAME( 'racfAutomatic'  'racfAutomatic' )
 ACCESS-CLASS sensitive LENGTH 20 )
ibmattributetypes=( 1.3.18.0.2.4.3435 DBNAME( 'racfStdataTrusted'  'racfStdataTrusted' )
 ACCESS-CLASS sensitive LENGTH 3 )
ibmattributetypes=( 1.3.18.0.2.4.3436 DBNAME( 'racfUpdateAccessC'  'racfUpdateAccessC' )
 ACCESS-CLASS sensitive LENGTH 15 )
ibmattributetypes=( 1.3.18.0.2.4.3437 DBNAME( 'racfVolumeList'  'racfVolumeList' )
 ACCESS-CLASS sensitive LENGTH 50 )
ibmattributetypes=( 1.3.18.0.2.4.3438 DBNAME( 'racfTimeZone'  'racfTimeZone' )
 ACCESS-CLASS sensitive LENGTH 10 )
ibmattributetypes=( 1.3.18.0.2.4.3439 DBNAME( 'racfStdataGroup'  'racfStdataGroup' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3440 DBNAME( 'racfStdataPrivile'  'racfStdataPrivile' )
 ACCESS-CLASS sensitive LENGTH 3 )
ibmattributetypes=( 1.3.18.0.2.4.3441 DBNAME( 'racfSsignonKeyEnc'  'racfSsignonKeyEnc' )
 ACCESS-CLASS critical LENGTH 16 )
ibmattributetypes=( 1.3.18.0.2.4.3442 DBNAME( 'racfSsignonKeyMas'  'racfSsignonKeyMas' )
 ACCESS-CLASS critical LENGTH 16 )
ibmattributetypes=( 1.3.18.0.2.4.3443 DBNAME( 'racfSessionLock'  'racfSessionLock' )
 ACCESS-CLASS sensitive LENGTH 10 )
ibmattributetypes=( 1.3.18.0.2.4.3444 DBNAME( 'racfSetroptsAttri'  'racfSetroptsAttri' )
 ACCESS-CLASS sensitive LENGTH 25 )
ibmattributetypes=( 1.3.18.0.2.4.3445 DBNAME( 'racfSigverFailLoa'  'racfSigverFailLoa' )
 ACCESS-CLASS sensitive LENGTH 15 )
ibmattributetypes=( 1.3.18.0.2.4.3446 DBNAME( 'racfSigverSigAudi'  'racfSigverSigAudi' )
 ACCESS-CLASS sensitive LENGTH 15 )
ibmattributetypes=( 1.3.18.0.2.4.3447 DBNAME( 'racfSigverSigRequ'  'racfSigverSigRequ' )
 ACCESS-CLASS sensitive LENGTH 3 )
ibmattributetypes=( 1.3.18.0.2.4.3448 DBNAME( 'racfSessionSessKe'  'racfSessionSessKe' )
 ACCESS-CLASS sensitive LENGTH 20 )
ibmattributetypes=( 1.3.18.0.2.4.3449 DBNAME( 'racfResourceAttri'  'racfResourceAttri' )
 ACCESS-CLASS sensitive LENGTH 25 )
ibmattributetypes=( 1.3.18.0.2.4.3450 DBNAME( 'racfResourceAudit'  'racfResourceAudit' )
 ACCESS-CLASS sensitive LENGTH 50 )
ibmattributetypes=( 1.3.18.0.2.4.3451 DBNAME( 'racfResourceGloba'  'racfResourceGloba' )
 ACCESS-CLASS sensitive LENGTH 50 )
ibmattributetypes=( 1.3.18.0.2.4.3452 DBNAME( 'racfReadAccessCou'  'racfReadAccessCou' )
 ACCESS-CLASS sensitive LENGTH 15 )
ibmattributetypes=( 1.3.18.0.2.4.3453 DBNAME( 'racfSessionConvSe'  'racfSessionConvSe' )
 ACCESS-CLASS sensitive LENGTH 15 )
ibmattributetypes=( 1.3.18.0.2.4.3454 DBNAME( 'racfSessionInterv'  'racfSessionInterv' )
 ACCESS-CLASS sensitive LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.3455 DBNAME( 'racfLogOptionsSuc'  'racfLogOptionsSuc' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3456 DBNAME( 'racfMemberList'  'racfMemberList' )
 ACCESS-CLASS sensitive LENGTH 512 )
ibmattributetypes=( 1.3.18.0.2.4.3457 DBNAME( 'racfNotify'  'racfNotify' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3458 DBNAME( 'racfLogOptionsNev'  'racfLogOptionsNev' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3459 DBNAME( 'racfRacList'  'racfRacList' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3460 DBNAME( 'racfLogOptionsFai'  'racfLogOptionsFai' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3461 DBNAME( 'racfLastReference'  'racfLastReference' )
 ACCESS-CLASS sensitive LENGTH 30 )
```

```
ibmattributetypes=( 1.3.18.0.2.4.3462 DBNAME( 'racfLastChangeDat'  'racfLastChangeDat' )
 ACCESS-CLASS sensitive LENGTH 30 )
ibmattributetypes=( 1.3.18.0.2.4.3463 DBNAME( 'racfLevel'  'racfLevel' )
 ACCESS-CLASS sensitive LENGTH 3 )
ibmattributetypes=( 1.3.18.0.2.4.3464 DBNAME( 'racfLogOptionsAlw'  'racfLogOptionsAlw' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3465 DBNAME( 'racfLogOptionsDef'  'racfLogOptionsDef' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3466 DBNAME( 'racfKerbPassword'  'racfKerbPassword' )
 ACCESS-CLASS critical LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3467 DBNAME( 'racfIctxDoMap'  'racfIctxDoMap' )
 ACCESS-CLASS sensitive LENGTH 3 )
ibmattributetypes=( 1.3.18.0.2.4.3468 DBNAME( 'racfIctxMapRequir'  'racfIctxMapRequir' )
 ACCESS-CLASS sensitive LENGTH 3 )
ibmattributetypes=( 1.3.18.0.2.4.3469 DBNAME( 'racfIctxUseMap'  'racfIctxUseMap' )
 ACCESS-CLASS sensitive LENGTH 3 )
ibmattributetypes=( 1.3.18.0.2.4.3470 DBNAME( 'racfKerbDefaultTi'  'racfKerbDefaultTi' )
 ACCESS-CLASS sensitive LENGTH 15 )
ibmattributetypes=( 1.3.18.0.2.4.3471 DBNAME( 'racfKerbMinTicket'  'racfKerbMinTicket' )
 ACCESS-CLASS sensitive LENGTH 15 )
ibmattributetypes=( 1.3.18.0.2.4.3472 DBNAME( 'racfIctxMappingTi'  'racfIctxMappingTi' )
 ACCESS-CLASS sensitive LENGTH 4 )
ibmattributetypes=( 1.3.18.0.2.4.3473 DBNAME( 'racfGlobal'  'racfGlobal' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3474 DBNAME( 'racfGeneric'  'racfGeneric' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3475 DBNAME( 'racfEimX509Regist'  'racfEimX509Regist' )
 ACCESS-CLASS sensitive LENGTH 255 )
ibmattributetypes=( 1.3.18.0.2.4.3476 DBNAME( 'racfEimOptions'  'racfEimOptions' )
 ACCESS-CLASS sensitive LENGTH 15 )
ibmattributetypes=( 1.3.18.0.2.4.3477 DBNAME( 'racfDlfdataJobNam'  'racfDlfdataJobNam' )
 ACCESS-CLASS sensitive LENGTH 50 )
ibmattributetypes=( 1.3.18.0.2.4.3478 DBNAME( 'racfDlfdataRetain'  'racfDlfdataRetain' )
 ACCESS-CLASS sensitive LENGTH 3 )
ibmattributetypes=( 1.3.18.0.2.4.3479 DBNAME( 'racfEimDomainDn'  'racfEimDomainDn' )
 ACCESS-CLASS sensitive LENGTH 1023 )
ibmattributetypes=( 1.3.18.0.2.4.3480 DBNAME( 'racfEimKerbRegist'  'racfEimKerbRegist' )
 ACCESS-CLASS sensitive LENGTH 255 )
ibmattributetypes=( 1.3.18.0.2.4.3481 DBNAME( 'racfEimLocalRegis'  'racfEimLocalRegis' )
 ACCESS-CLASS sensitive LENGTH 255 )
ibmattributetypes=( 1.3.18.0.2.4.3482 DBNAME( 'racfCopyProfileFr'  'racfCopyProfileFr' )
 ACCESS-CLASS sensitive LENGTH 512 )
ibmattributetypes=( 1.3.18.0.2.4.3483 DBNAME( 'racfCfdefType'  'racfCfdefType' )
 ACCESS-CLASS sensitive LENGTH 10 )
ibmattributetypes=( 1.3.18.0.2.4.3484 DBNAME( 'racfClassAct'  'racfClassAct' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3485 DBNAME( 'racfCfdefMinValue'  'racfCfdefMinValue' )
 ACCESS-CLASS sensitive LENGTH 20 )
ibmattributetypes=( 1.3.18.0.2.4.3486 DBNAME( 'racfCfdefHelp'  'racfCfdefHelp' )
 ACCESS-CLASS sensitive LENGTH 256 )
ibmattributetypes=( 1.3.18.0.2.4.3487 DBNAME( 'racfCfdefMaxLengt'  'racfCfdefMaxLengt' )
 ACCESS-CLASS sensitive LENGTH 10 )
ibmattributetypes=( 1.3.18.0.2.4.3488 DBNAME( 'racfCdtinfoSignal'  'racfCdtinfoSignal' )
 ACCESS-CLASS sensitive LENGTH 3 )
ibmattributetypes=( 1.3.18.0.2.4.3489 DBNAME( 'racfCdtinfoRacLis'  'racfCdtinfoRacLis' )
 ACCESS-CLASS sensitive LENGTH 20 )
ibmattributetypes=( 1.3.18.0.2.4.3490 DBNAME( 'racfCdtinfoPosit'  'racfCdtinfoPosit' )
 ACCESS-CLASS sensitive LENGTH 10 )
ibmattributetypes=( 1.3.18.0.2.4.3491 DBNAME( 'racfCdtinfoProfil'  'racfCdtinfoProfil' )
 ACCESS-CLASS sensitive LENGTH 3 )
ibmattributetypes=( 1.3.18.0.2.4.3492 DBNAME( 'racfCdtinfoOperat'  'racfCdtinfoOperat' )
 ACCESS-CLASS sensitive LENGTH 3 )
ibmattributetypes=( 1.3.18.0.2.4.3493 DBNAME( 'racfCdtinfoSecLab'  'racfCdtinfoSecLab' )
 ACCESS-CLASS sensitive LENGTH 3 )
ibmattributetypes=( 1.3.18.0.2.4.3494 DBNAME( 'racfCdtinfoMacPro'  'racfCdtinfoMacPro' )
 ACCESS-CLASS sensitive LENGTH 10 )
ibmattributetypes=( 1.3.18.0.2.4.3495 DBNAME( 'racfCdtinfoMaxLnx'  'racfCdtinfoMaxLnx' )
 ACCESS-CLASS sensitive LENGTH 3 )
ibmattributetypes=( 1.3.18.0.2.4.3496 DBNAME( 'racfCdtinfoMaxLen'  'racfCdtinfoMaxLen' )
 ACCESS-CLASS sensitive LENGTH 3 )
ibmattributetypes=( 1.3.18.0.2.4.3497 DBNAME( 'racfCdtinfoMember'  'racfCdtinfoMember' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3498 DBNAME( 'racfCdtinfoGroup'  'racfCdtinfoGroup' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3499 DBNAME( 'racfCdtinfoDfltrc'  'racfCdtinfoDfltrc' )
 ACCESS-CLASS sensitive LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.3500 DBNAME( 'racfCdtinfoCase'  'racfCdtinfoCase' )
 ACCESS-CLASS sensitive LENGTH 10 )
ibmattributetypes=( 1.3.18.0.2.4.3501 DBNAME( 'racfCdtinfoDefaul'  'racfCdtinfoDefaul' )
 ACCESS-CLASS sensitive LENGTH 10 )
ibmattributetypes=( 1.3.18.0.2.4.3502 DBNAME( 'racfCdtinfoGeneri'  'racfCdtinfoGeneri' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3503 DBNAME( 'racfCdtinfoGenLis'  'racfCdtinfoGenLis' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3504 DBNAME( 'racfAudit'  'racfAudit' )
 ACCESS-CLASS sensitive LENGTH 8 )
ibmattributetypes=( 1.3.18.0.2.4.3505 DBNAME( 'racfAlterAccessCo'  'racfAlterAccessCo' )
 ACCESS-CLASS sensitive LENGTH 16 )
ibmattributetypes=( 1.3.18.0.2.4.3506 DBNAME( 'profileName'  'profileName' )
 ACCESS-CLASS sensitive LENGTH 246 EQUALITY )
ibmattributetypes=( 1.3.18.0.2.4.3508 DBNAME( 'replicationWaitOn'  'replicationWaitOn' )
 ACCESS-CLASS normal LENGTH 5 )
ibmattributetypes=( 1.3.18.0.2.4.3509 DBNAME( 'slapdReplVersion'  'slapdReplVersion' )
 ACCESS-CLASS normal LENGTH 5)
ibmattributetypes=( 1.3.18.0.2.4.3511 DBNAME( 'racfIcsfAsymUsage'  'racfIcsfAsymUsage' )
 ACCESS-CLASS sensitive LENGTH 20 )
ibmattributetypes=( 1.3.18.0.2.4.3513 DBNAME( 'racfIcsfSymExport'  'racfIcsfSymExport' )
 ACCESS-CLASS sensitive LENGTH 20 )
ibmattributetypes=( 1.3.18.0.2.4.3514 DBNAME( 'racfIcsfSymExpCer'  'racfIcsfSymExpCer' )
 ACCESS-CLASS sensitive LENGTH 100 )
ibmattributetypes=( 1.3.18.0.2.4.3515 DBNAME( 'racfIcsfSymExpKey'  'racfIcsfSymExpKey' )
 ACCESS-CLASS sensitive LENGTH 64 )
ibmattributetypes=( 1.3.18.0.2.4.3523 DBNAME( 'filterBindMechani'  'filterBindMechani' )
 ACCESS-CLASS normal LENGTH 20 )
ibmattributetypes=( 1.3.18.0.2.4.3524 DBNAME( 'filterConnectionE'  'filterConnectionE' )
 ACCESS-CLASS normal LENGTH 5 )
```

```
ibmattributetypes=( 1.3.18.0.2.4.3526 DBNAME( 'filterDayOfWeek'  'filterDayOfWeek' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.3527 DBNAME( 'filterIP'  'filterIP' )
 ACCESS-CLASS normal LENGTH 240 )
ibmattributetypes=( 1.3.18.0.2.4.3528 DBNAME( 'filterSubject'  'filterSubject' )
 ACCESS-CLASS normal LENGTH 1000 )
ibmattributetypes=( 1.3.18.0.2.4.3529 DBNAME( 'filterTimeOfDay'  'filterTimeOfDay' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.3559 DBNAME ( 'SAFSecurityDomain' 'SAFSecurityDomain' )
 ACCESS-CLASS critical LENGTH 246 )
ibmattributetypes=( 1.3.18.0.2.4.3560 DBNAME( 'racfKerbCheckAddr'  'racfKerbCheckAddr' )
 ACCESS-CLASS sensitive LENGTH 3 )
ibmattributetypes=( 1.3.18.0.2.4.3656 DBNAME( 'ibm-slapdServerCompatibilityLevel'
 'ibm-slapdServerCompatibilityLevel' )
 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.454 DBNAME ( 'pwMaxRepChars' 'pwMaxRepChars' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.469 DBNAME ( 'pwMinOtherChars' 'pwMinOtherChars' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.470 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.473 DBNAME ( 'pwMinAlphaChars' 'pwMinAlphaChars' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.499 DBNAME ( 'pwMinDiffChars' 'pwMinDiffChars' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.18.0.2.4.826 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.827 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.828 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.829 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.830 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.831 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.6.1.1.4 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.1.5 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.1466.101.120.13 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.1466.101.120.14 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.1466.101.120.15 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.1466.101.120.16 ACCESS-CLASS system )
ibmattributetypes=( 1.3.6.1.4.1.1466.101.120.5 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.1466.101.120.6 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.1466.101.120.7 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.1 DBNAME ( 'pwdAttribute' 'pwdAttribute' )
 ACCESS-CLASS normal LENGTH 64 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.10 DBNAME ( 'pwdLockoutDuratio' 'pwdLockoutDuratio' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.11 DBNAME ( 'pwdMaxFailure' 'pwdMaxFailure' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.12 DBNAME ( 'pwdFailCntInt' 'pwdFailCntInt' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.13 DBNAME ( 'pwdMustChange' 'pwdMustChange' )
 ACCESS-CLASS normal LENGTH 5 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.14 DBNAME ( 'pwdAllowChange' 'pwdAllowChange' )
 ACCESS-CLASS normal LENGTH 5 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.15 DBNAME ( 'pwdSafeModify' 'pwdSafeModify' )
 ACCESS-CLASS normal LENGTH 5 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.16 DBNAME ( 'pwdChangedTime' 'pwdChangedTime' )
 ACCESS-CLASS critical LENGTH 30 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.17 DBNAME ( 'pwdAccLockTime' 'pwdAccLockTime' )
 ACCESS-CLASS critical LENGTH 30 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.18 DBNAME ( 'pwdExpireWarned' 'pwdExpireWarned' )
 ACCESS-CLASS critical LENGTH 30 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.19 DBNAME ( 'pwdFailureTime' 'pwdFailureTime' )
 ACCESS-CLASS critical LENGTH 30 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.2 DBNAME ( 'pwdMinAge' 'pwdMinAge' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.20 DBNAME ( 'pwdHistory' 'pwdHistory' )
 ACCESS-CLASS critical LENGTH 1024 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.21 DBNAME ( 'pwdGraceUseTime' 'pwdGraceUseTime' )
 ACCESS-CLASS critical LENGTH 30 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.22 DBNAME ( 'pwdReset' 'pwdReset' )
 ACCESS-CLASS critical LENGTH 5 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.3 DBNAME ( 'pwdMaxAge' 'pwdMaxAge' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.4 DBNAME ( 'pwdInHistory' 'pwdInHistory' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.5 DBNAME ( 'pwdCheckSyntax' 'pwdCheckSyntax' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.6 DBNAME ( 'pwdMinLength' 'pwdMinLength' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.7 DBNAME ( 'pwdExpireWarning' 'pwdExpireWarning' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.8 DBNAME ( 'pwdGraceLoginLimi' 'pwdGraceLoginLimi' )
 ACCESS-CLASS normal LENGTH 11 )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.9 DBNAME ( 'pwdLockout' 'pwdLockout' )
 ACCESS-CLASS normal LENGTH 5 )
ibmattributetypes=( 2.16.840.1.113730.3.1.10 ACCESS-CLASS normal )
ibmattributetypes=( 2.16.840.1.113730.3.1.11 ACCESS-CLASS normal )
ibmattributetypes=( 2.16.840.1.113730.3.1.198 ACCESS-CLASS normal )
ibmattributetypes=( 2.16.840.1.113730.3.1.34 ACCESS-CLASS normal )
ibmattributetypes=( 2.16.840.1.113730.3.1.35 ACCESS-CLASS normal )
ibmattributetypes=( 2.16.840.1.113730.3.1.5 ACCESS-CLASS normal )
ibmattributetypes=( 2.16.840.1.113730.3.1.6 ACCESS-CLASS normal )
ibmattributetypes=( 2.16.840.1.113730.3.1.7 ACCESS-CLASS normal )
ibmattributetypes=( 2.16.840.1.113730.3.1.77 ACCESS-CLASS normal )
ibmattributetypes=( 2.16.840.1.113730.3.1.8 ACCESS-CLASS sensitive )
ibmattributetypes=( 2.16.840.1.113730.3.1.9 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.18.1 ACCESS-CLASS system )
ibmattributetypes=( 2.5.18.10 ACCESS-CLASS system )
ibmattributetypes=( 2.5.18.2 ACCESS-CLASS system )
ibmattributetypes=( 2.5.18.3 ACCESS-CLASS system )
ibmattributetypes=( 2.5.18.4 ACCESS-CLASS system )
ibmattributetypes=( 2.5.18.6 ACCESS-CLASS system )
ibmattributetypes=( 2.5.18.9 DBNAME( 'hasSubordinates'  'hasSubordinates' )
 ACCESS-CLASS system LENGTH 5 )
ibmattributetypes=( 2.5.21.1 ACCESS-CLASS system )
ibmattributetypes=( 2.5.21.2 ACCESS-CLASS system )
ibmattributetypes=( 2.5.21.4 ACCESS-CLASS system )
ibmattributetypes=( 2.5.21.5 ACCESS-CLASS system )
```

```
ibmattributetypes=( 2.5.21.6 ACCESS-CLASS system )
ibmattributetypes=( 2.5.21.7 ACCESS-CLASS system )
ibmattributetypes=( 2.5.21.8 ACCESS-CLASS system )
ibmattributetypes=( 2.5.4.0 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.1 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.10 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.11 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.13 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.15 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.3 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.31 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.32 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.34 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.35 ACCESS-CLASS critical )
ibmattributetypes=( 2.5.4.41 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.49 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.50 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.6 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.7 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.8 ACCESS-CLASS normal )
objectclasses=( 1.3.18.0.2.6.174  NAME ( 'ibmSubschema' ) AUXILIARY SUP ( subschema )
 MAY ( ibmAttributeTypes ) )
objectclasses=( 1.3.18.0.2.6.241  NAME ( 'ibm-securityIdentities' ) DESC 'Defines the security
 identities of a user' AUXILIARY SUP ( top ) MAY ( altSecurityIdentities $ userPrincipalName ) )
objectclasses=( 1.3.18.0.2.6.248  NAME ( 'racfLNotesSegment' ) DESC 'Represents the z/OS
 LNOTES segment information in a RACF USER profile' AUXILIARY SUP ( top ) MAY ( racfLNotesShortName ) )
objectclasses=( 1.3.18.0.2.6.249  NAME ( 'racfNDSSegment' ) DESC 'Represents the z/OS
 NDS segment information in a RACF USER profile' AUXILIARY SUP ( top ) MAY ( racfNDSUserName ) )
objectclasses=( 1.3.18.0.2.6.259  NAME ( 'racfConnect' ) DESC 'RACF Connect'
 STRUCTURAL SUP ( top ) MUST ( racfGroupId $ racfUserid ) MAY ( racfConnectAttributes
 $ racfConnectAuthDate $ racfConnectCount $ racfConnectGroupAuthority $ racfConnectGroupUACC
 $ racfConnectLastConnect $ racfConnectOwner $ racfConnectResumeDate $ racfConnectRevokeDate ) )
objectclasses=( 1.3.18.0.2.6.260  NAME ( 'racfKerberosInfo' ) DESC 'Kerberos information
 for RACF' AUXILIARY SUP ( top ) MAY ( krbPrincipalName $ maxTicketAge $ racfCurKeyVersion
 $ racfEncryptType $ racfKerbKeyFrom ) )
objectclasses=( 1.3.18.0.2.6.261  NAME ( 'krbAlias' ) DESC 'Kerberos aliases' AUXILIARY SUP ( top )
 MAY ( krbAliasedObjectName $ krbHintAliases ) )
objectclasses=( 1.3.18.0.2.6.262  NAME ( 'ibm-changeLog' ) DESC 'IBM extension to changeLogEntry
 object class' AUXILIARY SUP ( top ) MAY ( ibm-changeInitiatorsName ) )
objectclasses=( 1.3.18.0.2.6.263  NAME ( 'krbRealm-V2' ) DESC 'Represents a Kerberos realm'
 STRUCTURAL SUP ( top ) MUST ( krbPrincSubtree $ krbRealmName-V2 ) )
objectclasses=( 1.3.18.0.2.6.264  NAME ( 'ibm-nativeAuthentication' ) DESC 'Use native
 security manager for authentication' AUXILIARY SUP ( top ) MUST ( ibm-nativeId ) )
objectclasses=( 1.3.18.0.2.6.267  NAME ( 'racfProxySegment' ) DESC 'RACF Proxy segment'
 AUXILIARY SUP ( top ) MAY ( racfLDAPBindDN $ racfLDAPBindPw $ racfLDAPHost ) )
objectclasses=( 1.3.18.0.2.6.28  NAME ( 'container' ) DESC 'An object that can contain
 other objects' STRUCTURAL SUP ( top ) MUST ( cn ) )
objectclasses=( 1.3.18.0.2.6.447  NAME ( 'racfEIMSegment' ) DESC 'RACF EIM segment'
 AUXILIARY SUP ( top ) MAY ( racfLDAPProf ) )
objectclasses=( 1.3.18.0.2.6.448  NAME ( 'ibm-nestedGroup' ) DESC 'Allow subgroups to be
 nested within parent group' AUXILIARY SUP ( top ) MAY ( ibm-memberGroup ) )
objectclasses=( 1.3.18.0.2.6.449  NAME ( 'ibm-dynamicGroup' ) DESC 'Used to create a
 hybrid group with both static and dynamic members' AUXILIARY SUP ( top ) MAY ( memberURL ) )
objectclasses=( 1.3.18.0.2.6.451  NAME ( 'ibm-staticGroup' ) DESC 'Used to create a
 hybrid group with both static and dynamic members' AUXILIARY SUP ( top ) MAY ( member ) )
objectclasses=( 1.3.18.0.2.6.476  NAME ( 'ibm-replicaGroup' ) DESC 'Represents a
 collection of servers participating in replication' STRUCTURAL SUP ( top ) MUST ( ibm-replicaGroup )
 MAY ( description ) )
objectclasses=( 1.3.18.0.2.6.477  NAME ( 'ibm-replicaSubentry' ) DESC 'Represents a single
 server participating in replication within a given subtree' STRUCTURAL SUP ( top ) MUST ( cn
 $ ibm-replicaServerId $ ibm-replicationServerIsMaster ) MAY ( description ) )
objectclasses=( 1.3.18.0.2.6.478  NAME ( 'ibm-replicationCredentialsExternal' )
 DESC 'SSL/TLS EXTERNAL credential information' STRUCTURAL SUP ( ibm-replicationCredentials )
 MAY ( ibm-replicaKeyfile $ ibm-replicaKeylabel $ ibm-replicaKeypwd $ ibm-replicaPKCS11Enabled ) )
objectclasses=( 1.3.18.0.2.6.479  NAME ( 'ibm-replicationCredentialsKerberos' )
 DESC 'Kerberos credential information' STRUCTURAL SUP ( ibm-replicationCredentials )
 MAY ( replicaBindDN $ replicaCredentials ) )
objectclasses=( 1.3.18.0.2.6.480  NAME ( 'ibm-replicationDailySchedule' )
 DESC 'Defines single day schedule for replication' STRUCTURAL SUP ( top )
 MAY ( cn $ description $ ibm-replDailySchedName $ ibm-replicationBatchStart
 $ ibm-replicationImmediateStart $ ibm-replicationTimesUTC ) )
objectclasses=( 1.3.18.0.2.6.481  NAME ( 'ibm-replicationCredentialsSimple' )
 DESC 'Simple bind credential information' STRUCTURAL SUP ( ibm-replicationCredentials )
 MUST ( replicaBindDN $ replicaCredentials ) )
objectclasses=( 1.3.18.0.2.6.482  NAME ( 'ibm-replicationWeeklySchedule' )
 DESC 'Defines weekly schedule for replication' STRUCTURAL SUP ( top ) MAY ( cn
 $ description $ ibm-replWeeklySchedName $ ibm-scheduleFriday $ ibm-scheduleMonday
 $ ibm-scheduleSaturday $ ibm-scheduleSunday $ ibm-scheduleThursday
 $ ibm-scheduleTuesday $ ibm-scheduleWednesday ) )
objectclasses=( 1.3.18.0.2.6.483  NAME ( 'ibm-replicationAgreement' )
 DESC 'Represents replication of a given subtree from a server to the consumer
 identified in this object' STRUCTURAL SUP ( top ) MUST ( cn $ ibm-replicaConsumerId
 $ ibm-replicaCredentialsDN $ ibm-replicaURL ) MAY ( description $ ibm-replicaConsumerConnections
 $ ibm-replicaMethod $ ibm-replicaScheduleDN $ ibm-replicationCreateMissingEntries
 $ ibm-replicationExcludedCapability $ ibm-replicationFilterDN $ ibm-replicationOnHold
 $ ibm-replicationWaitOnDependency ) )
objectclasses=( 1.3.18.0.2.6.484  NAME ( 'ibm-replicationContext' ) DESC 'Indicates
 that this entry is the root of a replicated subtree' AUXILIARY SUP ( top )
 MAY ( ibm-replicaReferralURL ) )
objectclasses=( 1.3.18.0.2.6.486  NAME ( 'ibm-slapdConfigEntry' ) DESC 'ibm slapd
 config entry' ABSTRACT SUP ( top ) MUST ( cn ) MAY ( ibm-slapdInvalidLine
 $ ibm-slapdMigrationInfo ) )
objectclasses=( 1.3.18.0.2.6.488  NAME ( 'ibm-slapdSupplier' ) DESC 'Contains
 bind credentials used by a replication supplier server to update the specified
 subtree on this consumer server. Use of this object class overrides the default
 bind credentials specified in an ibm-slapdReplication object.' STRUCTURAL SUP
 ( ibm-slapdConfigEntry $ top ) MUST ( cn $ ibm-slapdMasterDN
 $ ibm-slapdReplicaSubtree ) MAY ( ibm-slapdMasterPW ) )
objectclasses=( 1.3.18.0.2.6.496  NAME ( 'ibm-slapdReplication' )
 DESC 'Contains the default bind credentials and master server referral URL.
 This is used when the server contains one or more replication contexts that
 are replicated to it by other servers.  This server may be acting as one of
 several masters or as a read only replica.  If the MasterDN is specified
 without the Master PW attribute, kerberos authentication is used.'
 STRUCTURAL SUP ( ibm-slapdConfigEntry $ top ) MUST ( cn ) MAY ( ibm-slapdMasterDN
 $ ibm-slapdMasterPW $ ibm-slapdMasterReferral $ ibm-slapdNoReplConflictResolution ) )
```

```
objectclasses=( 1.3.18.0.2.6.521  NAME ( 'ibm-replicationCredentials' )
 DESC 'Base class for all replication credential objects' ABSTRACT SUP ( top )
 MAY ( cn $ description $ ibm-replCredName ) )
objectclasses=( 1.3.18.0.2.6.524  NAME ( 'ibm-pwdPolicyExt' ) DESC 'Defines
 the ibm extension for the policy for pwdPolicy object class.' AUXILIARY SUP
 ( pwdPolicy ) MAY ( ibm-pwdPolicy $ passwordMinAlphaChars $ passwordMinOtherChars
 $ passwordMinDiffChars $ passwordMaxRepeatedChars $ ibm-pwdPolicyStartTime
 $ passwordMaxConsecutiveRepeatedChars ) )
objectclasses=( 1.3.18.0.2.6.55  NAME ( 'racfbase' ) DESC 'Represents the
 base of the Directory Information Tree that publishs information stored by
 the z/OS Security Server RACF service' STRUCTURAL SUP ( top ) MAY ( sysplex ) )
objectclasses=( 1.3.18.0.2.6.555  NAME ( 'ibm-replicaGateway' )
 DESC 'An auxiliary class attached to an ibm-replicaSubentry to indicate
 the associated server is acting as a gateway server.' AUXILIARY SUP ( top ) )
objectclasses=( 1.3.18.0.2.6.556  NAME ( 'ibm-slapdAdminGroupMember' )
 DESC 'A User belonging to the IBM Directory Server Administration Group.'
 STRUCTURAL SUP ( top $ ibm-slapdConfigEntry ) MUST ( ibm-slapdAdminDN
 $ ibm-slapdAdminRole ) MAY ( ibm-slapdKrbAdminDN $ ibm-slapdDigestAdminUser
 $ ibm-slapdAdminPW ) )
objectclasses=( 1.3.18.0.2.6.56  NAME ( 'racfProfileType' )
 DESC 'Represents a container below which individual RACF profile entries
 will be published' STRUCTURAL SUP ( top ) MUST ( profileType ) )
objectclasses=( 1.3.18.0.2.6.57  NAME ( 'racfBaseCommon' ) DESC 'Represents
 a commong base class for all RACF profiles' ABSTRACT SUP ( top ) MAY ( racfOwner
 $ racfInstallationData $ racfDatasetModel $ racfAuthorizationDate ) )
objectclasses=( 1.3.18.0.2.6.573  NAME ( 'ibm-searchLimits' ) DESC 'Can
 be used to define a group to have special search time limit and size limit.'
 AUXILIARY SUP ( top ) MUST ( cn $ ibm-searchTimeLimit $ ibm-searchSizeLimit ) )
objectclasses=( 1.3.18.0.2.6.58  NAME ( 'racfUser' ) DESC 'Represents a
 RACFUSER Profile entry' STRUCTURAL SUP ( racfBaseCommon ) MUST ( racfid )
 MAY ( racfAuthorizationDate $ racfAttributes $ racfPassword $ racfPasswordChangeDate
 $ racfPasswordEnvelope $ racfPasswordInterval $ racfProgrammerName $ racfDefaultGroup
 $ racfLastAccess $ racfSecurityLabel $ racfSecurityCategoryList $ racfRevokeDate
 $ racfResumeDate $ racfLogonDays $ racfLogonTime $ racfClassName $ racfConnectGroupName
 $ racfConnectGroupAuthority $ racfConnectGroupUACC $ racfSecurityLevel $ racfPassPhrase
 $ racfPassPhraseChangeDate $ racfHavePasswordEnvelope $ racfPassPhraseEnvelope
 $ racfHavePassPhraseEnvelope ) )
objectclasses=( 1.3.18.0.2.6.588  NAME ( 'ibm-slapdLogConfig' ) DESC 'Log management
 configuration.' AUXILIARY SUP ( top $ ibm-slapdConfigEntry ) MAY ( ibm-slapdLogMaxArchives
 $ ibm-slapdLogOptions $ ibm-slapdLogSizeThreshold $ ibm-slapdLogArchivePath
 $ ibm-slapdLog $ ibm-slapdLogMgmtStartTime $ ibm-slapdLogMgmtFrequency
 $ ibm-slapdLogEventFileEnabled $ ibm-slapdLogEventFilePath $ ibm-slapdLogEventFilePrefix
 $ ibm-slapdLogEventFileSizeThreshold $ ibm-slapdLogEventFileArchivePath
 $ ibm-slapdLogEventFileMaxArchives $ ibm-slapdLogEventFileOptions
 $ ibm-slapdLogCARSEnabled $ ibm-slapdLogCARSServer $ ibm-slapdLogCARSPort
 $ ibm-slapdLogCARSOptions $ ibm-slapdLogEventFormat $ ibm-slapdLogCachePath
 $ ibm-slapdAuditOperation ) )
objectclasses=( 1.3.18.0.2.6.59  NAME ( 'racfGroup' ) DESC 'Represents a RACF
 GROUP Profile entry' STRUCTURAL SUP ( racfBaseCommon ) MUST ( racfid )
 MAY ( racfSuperiorGroup $ racfGroupNoTermUAC $ racfSubGroupName $ racfGroupUserids
 $ racfGroupUniversal ) )
objectclasses=( 1.3.18.0.2.6.596  NAME ( 'ibm-slapdReplicationConfiguration' )
 DESC 'Used to configure replication for a supplier' STRUCTURAL SUP ( top ) MUST
 ( cn ) MAY ( description $ ibm-slapdMaxPendingChangesDisplayed $ ibm-slapdReplContextCacheSize
 $ ibm-slapdReplMaxErrors $ ibm-slapdReplConflictMaxEntrySize $ ibm-replicationOnHold
 $ ibm-slapdReplRestrictedAccess $ ibm-slapdEnableConflictResolutionForGroups
 $ibm-slapdReplicateSecurityAttributes ) )
objectclasses=( 1.3.18.0.2.6.60  NAME ( 'SAFDfpSegment' ) DESC 'Represents the SAF
 DFP portions of a RACF USER or GROUP profile' AUXILIARY SUP ( top ) MAY
 ( SAFDfpDataApplication $ SAFDfpDataClass $ SAFDfpManagementClass
 $ SAFDfpStorageClass ) )
objectclasses=( 1.3.18.0.2.6.61  NAME ( 'racfGroupOmvsSegment' )
 DESC 'Represents the z/OS OMVS Group information portion of a RACF GROUP
 profile' AUXILIARY SUP ( top ) MAY ( racfOmvsGroupId $ racfOmvsGroupIdKeyword ) )
objectclasses=( 1.3.18.0.2.6.62  NAME ( 'racfGroupOvmSegment' )
 DESC 'Represents the z/OS OVM Group information portion of a RACF GROUP
 profile' AUXILIARY SUP ( top ) MAY ( racfOvmGroupId ) )
objectclasses=( 1.3.18.0.2.6.63  NAME ( 'racfUserOmvsSegment' )
 DESC 'Represents the z/OS OMVS User information portion of a RACF USER
 profile' AUXILIARY SUP ( top ) MAY ( racfOmvsUid $ racfOmvsHome
 $ racfOmvsInitialProgram $ racfOmvsMaximumAddressSpaceSize $ racfOmvsMaximumCPUTime
 $ racfOmvsMaximumFilesPerProcess $ racfOmvsMaximumMemoryMapArea
 $ racfOmvsMaximumProcessesPerUID $ racfOmvsMaximumThreadsPerProcess
 $ racfOmvsMemoryLimit $ racfOmvsSharedMemoryMaximum $ racfOmvsUidKeyword ) )
objectclasses=( 1.3.18.0.2.6.630  NAME ( 'ibm-replicationFilter' )
 DESC 'This objectclass is used to define filters' STRUCTURAL SUP ( top )
 MUST ( cn $ ibm-replicationFilterAttr ) )
objectclasses=( 1.3.18.0.2.6.636  NAME ( 'ibm-pwdGroupAndIndividualPolicies' )
 DESC 'Contains attributes related to administering group and individual password policies.'
 AUXILIARY SUP ( top ) MUST ( ibm-pwdGroupAndIndividualEnabled ) )
objectclasses=( 1.3.18.0.2.6.64  NAME ( 'racfUserOvmSegment' )
 DESC 'Represents the z/OS OVM User information portion of a RACF USER profile'
 AUXILIARY SUP ( top ) MAY ( racfOvmUid $ racfOvmHome $ racfOvmInitialProgram
 $ racfOvmFileSystemRoot ) )
objectclasses=( 1.3.18.0.2.6.65  NAME ( 'SAFTsoSegment' ) DESC 'Represents the
 z/OS TSO information in a RACF USER profile' AUXILIARY SUP ( top ) MAY
 ( SAFAccountNumber $ SAFDestination $ SAFHoldClass $ SAFJobClass $ SAFMessageClass
 $ SAFDefaultLoginProc $ SAFLogonSize $ SAFMaximumRegionSize $ SAFDefaultSysoutClass
 $ SAFUserdata $ SAFDefaultUnit $ SAFTsoSecurityLabel $ SAFDefaultCommand ) )
objectclasses=( 1.3.18.0.2.6.655  NAME ( 'racfResource' ) DESC 'Provides naming
 attribute for a discrete or generic RACF resource profile in a class' STRUCTURAL SUP
 ( top ) MUST ( profileName ) )
objectclasses=( 1.3.18.0.2.6.656  NAME ( 'ibm-tdszTop' ) DESC 'Global configuration
 settings for IBM Directory Server on z/OS.' STRUCTURAL SUP ( top $ ibm-slapdConfigEntry )
 MUST ( cn ) MAY ( ibm-slapdServerId $ ibm-slapdMaxPendingChangesDisplayed
 $ ibm-slapdAdminGroupEnabled $ ibm-slapdSAFSecurityDomain ) )
objectclasses=( 1.3.18.0.2.6.66  NAME ( 'racfCicsSegment' ) DESC 'Represents
 the z/OS CICS information in a RACF USER profile' AUXILIARY SUP ( top )
 MAY ( racfOperatorClass $ racfOperatorIdentification $ racfOperatorPriority
 $ racfOperatorReSignon $ racfRslKey $ racfTerminalTimeout $ racfTslKey ) )
objectclasses=( 1.3.18.0.2.6.667  NAME ( 'ibm-slapdSAFAdminGroup' )
 DESC 'A group of users with SAF native IDs belonging to the IBM Directory Server
 Administration Group. The admin roles for each user are defined in the z/OS
 security manager.' STRUCTURAL SUP ( top $ ibm-slapdConfigEntry ) MUST ( cn ) MAY ( member ) )
objectclasses=( 1.3.18.0.2.6.67  NAME ( 'racfOperparmSegment' ) DESC 'Represents
 the z/OS Operator parameters in a RACF USER profile' AUXILIARY SUP ( top )
```

```
 MAY ( racfStorageKeyword $ racfAuthKeyword $ racfMformKeyword $ racfLevelKeyword
 $ racfMonitorKeyword $ racfRoutcodeKeyword $ racfLogCommandResponseKeyword $ racfMGIDKeyword
 $ racfDOMKeyword $ racfKEYKeyword $ racfCMDSYSKeyword $ racfUDKeyword $ racfMscopeSystems
 $ racfAltGroupKeyword $ racfAutoKeyword $ racfHcKeyword $ racfIntidsKeyword $ racfUnknidsKeyword ) )
objectclasses=( 1.3.18.0.2.6.68  NAME ( 'racfLanguageSegment' ) DESC 'Represents the
 z/OS language information in a RACF USER profile' AUXILIARY SUP ( top ) MAY ( racfPrimaryLanguage
 $ racfSecondaryLanguage ) )
objectclasses=( 1.3.18.0.2.6.69  NAME ( 'racfWorkAttrSegment' ) DESC 'Represents the
 z/OS work attributes information in a RACF USER profile' AUXILIARY SUP ( top )
 MAY ( racfWorkAttrUsername $ racfBuilding $ racfDepartment $ racfRoom $ racfAddressLine1
 $ racfAddressLine2 $ racfAddressLine3 $ racfAddressLine4 $ racfWorkAttrAccountNumber ) )
objectclasses=( 1.3.18.0.2.6.70  NAME ( 'racfNetviewSegment' ) DESC 'Represents the z/OS
 Netview information in a RACF USER profile' AUXILIARY SUP ( top ) MAY ( racfNetviewInitialCommand
 $ racfDefaultConsoleName $ racfCTLKeyword $ racfMSGRCVRKeyword $ racfNetviewOperatorClass
 $ racfDomains $ racfNGMFADMKeyword $ racfNGMFVSPNKeyword ) )
objectclasses=( 1.3.18.0.2.6.71  NAME ( 'racfDCESegment' ) DESC 'Represents the z/OS DCE
 information in a RACF USER profile' AUXILIARY SUP ( top ) MAY ( racfDCEAutoLogin $ racfDCEHomeCell
 $ racfDCEHomeCellUUID $ racfDCEPrincipal $ racfDCEUUID ) )
objectclasses=( 1.3.18.0.2.6.72  NAME ( 'replicaObject' ) DESC 'Represents information about a
 directory server replica' STRUCTURAL SUP ( top ) MUST ( cn $ replicaBindDN $ replicaHost
 $ replicaCredentials ) MAY ( description $ seeAlso $ replicaPort $ replicaBindMethod $ replicaUseSSL
 $ replicaUpdateTimeInterval ) )
objectclasses=( 1.3.18.0.2.6.74  NAME ( 'aliasObject' ) DESC 'Defines an alias for a directory entry'
 AUXILIARY SUP ( top ) MUST ( aliasedObjectName ) )
objectclasses=( 1.3.18.0.2.6.75  NAME ( 'accessGroup' ) DESC 'Group used for access control'
 STRUCTURAL SUP ( top ) MUST ( cn ) MAY ( member $ businessCategory $ seeAlso $ owner $ ou $ o
 $ description ) )
objectclasses=( 1.3.18.0.2.6.76  NAME ( 'accessRole' ) DESC 'Role used for access control'
 STRUCTURAL SUP ( top ) MUST ( cn ) MAY ( member $ businessCategory $ seeAlso $ owner $ ou $ o
 $ description ) )
objectclasses=( 1.3.6.1.4.1.1466.101.120.111  NAME ( 'extensibleObject' )
 DESC 'Permits the entry to hold any attribute type defined in the schema' AUXILIARY SUP ( top ) )
objectclasses=( 1.3.6.1.4.1.42.2.27.8.2.1  NAME ( 'pwdPolicy' ) DESC 'Defines a policy for
 password management' AUXILIARY SUP ( top ) MUST ( pwdAttribute ) MAY ( pwdMinAge $ pwdMaxAge
 $ pwdInHistory $ pwdCheckSyntax $ pwdMinLength $ pwdExpireWarning $ pwdGraceLoginLimit $ pwdLockout
 $ pwdLockoutDuration $ pwdMaxFailure $ pwdFailureCountInterval $ pwdMustChange $ pwdAllowUserChange
 $ pwdSafeModify ) )
objectclasses=( 2.16.840.1.113730.3.2.1  NAME ( 'changeLogEntry' )
 DESC 'Used to represent changes made to a directory server' STRUCTURAL SUP ( top ) MUST ( targetDN
 $ changeTime $ changeNumber $ changeType ) MAY ( modifiersName $ changes $ newRdn $ deleteOldRdn $ newSuperior ) )
objectclasses=( 2.16.840.1.113730.3.2.33  NAME ( 'groupOfURLs' ) DESC 'Represents a group of URLs'
 STRUCTURAL SUP ( top ) MUST ( cn ) MAY ( memberURL $ businessCategory $ description $ o $ ou $ owner $ seeAlso ) )
objectclasses=( 2.16.840.1.113730.3.2.6  NAME ( 'referral' ) DESC 'Defines a pointer to another server'
 STRUCTURAL SUP ( top ) MUST ( ref ) )
objectclasses=( 2.5.17.0  NAME ( 'subentry' ) STRUCTURAL SUP ( top ) MUST ( cn $ subtreeSpecification ) )
objectclasses=( 2.5.20.1  NAME ( 'subschema' ) AUXILIARY SUP ( top )
 MAY ( ditStructureRules $ nameForms $ ditContentRules $ objectClasses $ attributeTypes $
 matchingRules $ matchingRuleUse $ ldapSyntaxes ) )
objectclasses=( 2.5.6.0  NAME ( 'top' ) ABSTRACT MUST ( objectClass ) )
objectclasses=( 2.5.6.1  NAME ( 'alias' ) DESC 'Defines an alias for a directory entry' STRUCTURAL SUP ( top )
 MUST ( aliasedObjectName ) )
objectclasses=( 2.5.6.17  NAME ( 'groupOfUniqueNames' ) DESC 'Defines entries for a group of unique names' STRUCTURAL SUP ( top )
 MUST ( cn $ uniqueMember ) MAY ( businessCategory $ seeAlso $ owner $ ou $ o $ description ) )
objectclasses=( 2.5.6.9  NAME ( 'groupOfNames' ) DESC 'Defines entries for a group of names' STRUCTURAL SUP ( top )
 MUST ( cn $ member ) MAY ( businessCategory $ seeAlso $ owner $ ou $ o $ description ) )
```

Appendix B. SPUFI files

This topic shows the following SPUFI files:
- "The DSTDBMDB SPUFI file"
- "The TDBMMGRT SPUFI file" on page 669

Note: The same SPUFI files are used to generate both a TDBM and a GDBM (when DB2-based) database.

The DSTDBMDB SPUFI file

```
--*********************************************************************
--* Licensed Materials - Property of IBM
--* 5650-ZOS
--* Copyright IBM Corp.  2006, 2013
--*********************************************************************
--
-- Use the following statements to create your Directory Server DB2
-- database, tablespaces and indexes in SPUFI.
--
-- You will 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 -DB_NAME- to the name of the LDAP database name you want to
-- create.
-- Example: GLDDB
--
-- *************************
-- DataBase Owner Information
-- *************************
-- Change -DB_USERID- to the database owner id. This ID will be the
-- highlevel qualifier for the tables.  Be sure this value is updated to
-- match what is defined for dbUserid in the LDAP server configuration
-- file.
-- Example: GLDSRV
--
-- ******************************
-- LDAP Server User ID Information
-- ******************************
-- Change -LDAP_USERID- to the user ID for the LDAP server to run
-- under.
-- Example: GLDSRV
--
-- ******************************
-- Database Plan Name Information
-- ******************************
-- Change -DB_PLAN- to the DB2 CLI plan name.
-- Example: DSNACLI
--
-- **********************
-- Tablespace Information
-- **********************
--
--   ********************************************************************
--   NOTE: Refer to the DB2 manuals for a complete listing of valid
--         buffer pool names.
```

```
--   ********************************************************************
--
-- Change -ENTRYTS- to the LDAP entry tablespace name you want to
--     create.
--
-- Change -ENTRYTS_BP0- 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 suggested buffer pool name is BP0, which represents a buffer
--     pool size of 4K.
--
-- Change -LENTRYTS- to the LDAP long entry tablespace name you want
--     to create.
--
-- Change -LENTRYTS_BP0- 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 suggested buffer pool name is BP0, which represents a buffer
--     pool size of 4K.
--
-- Change -LATTRTS- to the LDAP long attribute tablespace name you
--     want to create.
--
-- Change -LATTRTS_BP0- 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 suggested buffer pool name is BP0, which represents a buffer
--     pool size of 4K.
--
-- Change -SEARCHTS- to the LDAP search tablespace name you want to
--     create.
--
-- Change -SEARCHTS_BP0- 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 suggested buffer pool name is BP0, which represents a buffer
--     pool size of 4K.
--
```

```
-- Change -MISCTS- to the LDAP miscellaneous tablespace name you
--     want to create.
--
-- Change -MISCTS_BP0- to the buffer pool name for the LDAP
--     miscellaneous tablespace.
--
--     The suggested buffer pool name is BP0, which represents a buffer
--     pool size of 4K.
--
-- Change -DESCTS- to the LDAP descendants tablespace name you want
--     to create.
--
-- Change -DESCTS_BP0- to the buffer pool name for the LDAP
--     descendants tablespace.
--
--     The suggested buffer pool name is BP0, which represents a buffer
--     pool size of 4K.
--
-- Change -REPTS- to the LDAP replica tablespace name you want to
--     create.
--
-- Change -REPTS_BP0- to the buffer pool name for the LDAP replica
--     tablespace.
--
--     The suggested buffer pool name is BP0, which represents a buffer
--     pool size of 4K.
--
-- ********************************
-- 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 -SEARCH_TRUNC_SIZE- to the search column trunc size you
-- determine best fits your attribute data.
--
--     The suggested size is 32.
--     The minimum size is 8
--
-- 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 -ENTRY_DN_TRUNC_SIZE- to the dn trunc size you determine best
-- fits your dn data.
--
--     The suggested size for a TDBM backend is 64.
--     The suggested size for a DB2 based GDBM backend is 32.
--     The minimum size is 8
```

```
--
-- Note: If you are going to use DB2 utilities to copy an existing TDBM
--       or GDBM database to the DB2 database created by this SPUFI script,
--       then you must use the dn trunc size of the DB2 database you are
--       copying.  Otherwise the DB2 copy utility will produce unreliable
--       results.
--
-- Change -ENTRY_DN_SIZE- to the maximum size of a DN.  This value
-- includes the null terminator, so the actual maximum length of a DN
-- will be one less than this value.
--
--     The suggested size is 512.
--
-- Note: The -ENTRY_DN_SIZE- value must be less than or equal to
--       the maximum size of the change DN column (CHNGDN) in the
--       replication table (DIR_REPENTRY).  -ENTRY_DN_SIZE- will be
--       used to define both column sizes.
--
-- ************************
-- Storage Group Information
-- ************************
-- Change -SYSDEFLT- 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.
--
-- NOTE: You need to remove the '--' from each line before you can run
--       these statements.
--
-- Change -DB_NAME- to the LDAP database name you want to delete.
--
-- Change the following names to the LDAP tablespace names you want to
-- delete:
--     -ENTRYTS-
--     -LENTRYTS-
--     -LATTRTS-
--     -SEARCHTS-
--     -MISCTS-
--     -DESCTS-
--     -REPTS-
-- ***************************************************************************
--DROP TABLESPACE -DB_NAME-.-ENTRYTS-;
--DROP TABLESPACE -DB_NAME-.-LENTRYTS-;
--DROP TABLESPACE -DB_NAME-.-LATTRTS-;
--DROP TABLESPACE -DB_NAME-.-SEARCHTS-;
--DROP TABLESPACE -DB_NAME-.-MISCTS-;
--DROP TABLESPACE -DB_NAME-.-REPTS-;
--DROP TABLESPACE -DB_NAME-.-DESCTS-;
--DROP DATABASE   -DB_NAME-;
--COMMIT;

-- ************************
-- Create the LDAP database
-- ************************
CREATE DATABASE -DB_NAME- STOGROUP -SYSDEFLT- CCSID EBCDIC;

-- *******************************
-- Create the LDAP entry tablespace
-- *******************************
CREATE TABLESPACE -ENTRYTS- IN -DB_NAME-
       USING STOGROUP -SYSDEFLT-
```

```
       PRIQTY 14400
       SECQTY 7200
       BUFFERPOOL -ENTRYTS_BP0-;

-- ************************************
-- Create the LDAP long entry tablespace
-- ************************************
CREATE TABLESPACE -LENTRYTS- IN -DB_NAME-
       USING STOGROUP -SYSDEFLT-
       PRIQTY 14400
       SECQTY 7200
       BUFFERPOOL -LENTRYTS_BP0-;

-- ***********************************
-- Create the LDAP long attr tablespace
-- ***********************************
CREATE TABLESPACE -LATTRTS- IN -DB_NAME-
       USING STOGROUP -SYSDEFLT-
       PRIQTY 14400
       SECQTY 7200
       BUFFERPOOL -LATTRTS_BP0-;

-- *********************************
-- Create the LDAP search tablespace
-- *********************************
CREATE TABLESPACE -SEARCHTS- IN -DB_NAME-
       USING STOGROUP -SYSDEFLT-
       PRIQTY 14400
       SECQTY 7200
       BUFFERPOOL -SEARCHTS_BP0-;

-- *******************************
-- Create the LDAP misc tablespace
-- *******************************
CREATE TABLESPACE -MISCTS- IN -DB_NAME-
       SEGSIZE 4
       USING STOGROUP -SYSDEFLT-
       PRIQTY 14400
       SECQTY 7200
       LOCKSIZE ROW
       BUFFERPOOL -MISCTS_BP0-;

-- **************************************
-- Create the LDAP descendants tablespace
-- **************************************
CREATE TABLESPACE -DESCTS- IN -DB_NAME-
       USING STOGROUP -SYSDEFLT-
       PRIQTY 14400
       SECQTY 7200
       BUFFERPOOL -DESCTS_BP0-;

-- *********************************
-- Create the LDAP replica tablespace
-- *********************************
CREATE TABLESPACE -REPTS- IN -DB_NAME-
       USING STOGROUP -SYSDEFLT-
       PRIQTY 14400
       SECQTY 7200
       BUFFERPOOL -REPTS_BP0-;

-- **********************
-- Create the DB2 tables
-- **********************

-- *************************
-- Create the DIR_ENTRY table
-- *************************
```

```
CREATE TABLE -DB_USERID-.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(-ENTRY_DN_TRUNC_SIZE-) FOR BIT DATA,
    DN                  VARCHAR(-ENTRY_DN_SIZE-) FOR BIT DATA,
    ENTRYDATA           LONG VARCHAR      FOR BIT DATA,
    PRIMARY KEY( EID ) )
IN -DB_NAME-.-ENTRYTS-;

-- ******************************
-- Create the DIR_LONGENTRY table
-- ******************************
CREATE TABLE -DB_USERID-.DIR_LONGENTRY (
    EID                 DECIMAL(15 , 0)   NOT NULL,
    SEQ                 INTEGER           NOT NULL,
    ENTRYDATA           LONG VARCHAR      FOR BIT DATA,
    PRIMARY KEY( EID, SEQ ) )
IN -DB_NAME-.-LENTRYTS-;

-- ******************************
-- Create the DIR_LONGATTR table
-- ******************************
CREATE TABLE -DB_USERID-.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 -DB_NAME-.-LATTRTS-;

-- ******************************
-- Create the DIR_MISC table
-- ******************************
CREATE TABLE -DB_USERID-.DIR_MISC (
    NEXT_EID            DECIMAL(15 , 0),
    NEXT_ATTR_ID        INTEGER,
    DB_VERSION          CHAR(10),
    DB_CREATE_VERSION   CHAR(10),
    SCHEMA_TIMESTAMP    TIMESTAMP,
    PARTITIONED_EID     CHAR(1) )
IN -DB_NAME-.-MISCTS-;

-- ***************************
-- Create the DIR_CACHE table
-- ***************************
CREATE TABLE -DB_USERID-.DIR_CACHE (
    CACHE_NAME          CHAR(25)          NOT NULL,
    MODIFY_TIMESTAMP    TIMESTAMP         NOT NULL,
    PRIMARY KEY( CACHE_NAME, MODIFY_TIMESTAMP ) )
IN -DB_NAME-.-MISCTS-;

-- ***************************
-- Create the DIR_ATTRID table
-- ***************************
CREATE TABLE -DB_USERID-.DIR_ATTRID (
    ATTR_ID             INTEGER,
    ATTR_NOID           VARCHAR(200)      NOT NULL,
    PRIMARY KEY( ATTR_NOID ) )
```

```
IN -DB_NAME-.-MISCTS-;

-- *************************
-- Create the DIR_DESC table
-- *************************
CREATE TABLE -DB_USERID-.DIR_DESC (
    DEID                DECIMAL(15 , 0)   NOT NULL,
    AEID                DECIMAL(15 , 0)   NOT NULL,
    PRIMARY KEY( DEID, AEID ) )
IN -DB_NAME-.-DESCTS-;

-- ****************************
-- Create the DIR_SEARCH table
-- ****************************
CREATE TABLE -DB_USERID-.DIR_SEARCH (
    EID                 DECIMAL(15 , 0)   NOT NULL,
    ATTR_ID             INTEGER           NOT NULL,
    VALUE               CHAR(-SEARCH_TRUNC_SIZE-) FOR BIT DATA,
    LVALUE              LONG VARCHAR      FOR BIT DATA )
IN -DB_NAME-.-SEARCHTS-;

-- *****************************
-- Create the DIR_REPLICA table
-- *****************************
CREATE TABLE -DB_USERID-.DIR_REPLICA (
    REPID               DECIMAL(15 , 0)   NOT NULL,
    CHNGID              DECIMAL(15 , 0)   NOT NULL,
    REPCAPS             INTEGER           NOT NULL,
    PRIMARY KEY( REPID ) )
IN -DB_NAME-.-MISCTS-;

-- ******************************
-- Create the DIR_REPENTRY table
-- ******************************
CREATE TABLE -DB_USERID-.DIR_REPENTRY (
    CHNGID              DECIMAL(15 , 0)   NOT NULL,
    CHNGFLAGS           INTEGER           NOT NULL,
    CHNGSIZE            INTEGER           NOT NULL,
    CHNGDN              VARCHAR(-ENTRY_DN_SIZE-) FOR BIT DATA,
    CHNGDATA            LONG VARCHAR      FOR BIT DATA,
    PRIMARY KEY( CHNGID ) )
IN -DB_NAME-.-REPTS-;

-- *********************************
-- Create the DIR_LONGREPENTRY table
-- *********************************
CREATE TABLE -DB_USERID-.DIR_LONGREPENTRY (
    CHNGID              DECIMAL(15 , 0)   NOT NULL,
    CHNGSEQ             INTEGER           NOT NULL,
    CHNGDATA            LONG VARCHAR      FOR BIT DATA,
    PRIMARY KEY( CHNGID, CHNGSEQ ) )
IN -DB_NAME-.-REPTS-;

-- ************************
-- Create the DIR_EID table
-- ************************
CREATE TABLE -DB_USERID-.DIR_EID (
    PARTID              DECIMAL(15 , 0)   NOT NULL,
    NEXT_EID            DECIMAL(15 , 0)   NOT NULL,
    MODIFY_TIMESTAMP    TIMESTAMP         NOT NULL,
    PRIMARY KEY( PARTID ) )
IN -DB_NAME-.-MISCTS-;

-- *******************************
-- Create the DIR_REPLSTATUS table
-- *******************************
CREATE TABLE -DB_USERID-.DIR_REPLSTATUS (
```

```
    AGREEMENTEID        DECIMAL(15 , 0)   NOT NULL,
    CHANGEID            INTEGER           NOT NULL,
    REPLICACAPS         CHAR(64)          FOR BIT DATA,
    PRIMARY KEY( AGREEMENTEID ) )
IN -DB_NAME-.-MISCTS-;

-- ******************************
-- Create the DIR_REPLCHANGE table
-- ******************************
CREATE TABLE -DB_USERID-.DIR_REPLCHANGE (
    CHANGEID            INTEGER           NOT NULL,
    CONTEXTEID          DECIMAL(15 , 0)   NOT NULL,
    CHANGESIZE          INTEGER           NOT NULL,
    CHANGEDATA          LONG VARCHAR      FOR BIT DATA,
    PRIMARY KEY( CHANGEID, CONTEXTEID ) )
IN -DB_NAME-.-REPTS-;

-- *********************************
-- Create the DIR_LONGREPLCHANGE table
-- *********************************
CREATE TABLE -DB_USERID-.DIR_LONGREPLCHANGE (
    CHANGEID            INTEGER           NOT NULL,
    CONTEXTEID          DECIMAL(15 , 0)   NOT NULL,
    CHANGESEQ           INTEGER           NOT NULL,
    CHANGEDATA          LONG VARCHAR      FOR BIT DATA,
    PRIMARY KEY( CHANGEID, CONTEXTEID, CHANGESEQ ) )
IN -DB_NAME-.-REPTS-;

-- ******************************
-- Create the DIR_REPLERROR table
-- ******************************
CREATE TABLE -DB_USERID-.DIR_REPLERROR (
    ERRORID             INTEGER           NOT NULL,
    AGREEMENTEID        DECIMAL(15 , 0)   NOT NULL,
    ERRORSIZE           INTEGER           NOT NULL,
    ERRORDATA           LONG VARCHAR      FOR BIT DATA,
    PRIMARY KEY( ERRORID ) )
IN -DB_NAME-.-REPTS-;

-- *********************************
-- Create the DIR_LONGREPERROR table
-- *********************************
CREATE TABLE -DB_USERID-.DIR_LONGREPLERROR (
    ERRORID             INTEGER           NOT NULL,
    AGREEMENTEID        DECIMAL(15 , 0)   NOT NULL,
    ERRORSEQ            INTEGER           NOT NULL,
    ERRORDATA           LONG VARCHAR      FOR BIT DATA,
    PRIMARY KEY( ERRORID, ERRORSEQ ) )
IN -DB_NAME-.-REPTS-;

-- *************************
-- 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 -DB_USERID-.DIR_ENTRYX0
    ON -DB_USERID-.DIR_ENTRY( EID )
    USING STOGROUP -SYSDEFLT-
    PRIQTY 720
```

```
    SECQTY 720
    CLUSTER
    DEFER YES;

CREATE INDEX -DB_USERID-.DIR_ENTRYX1
    ON -DB_USERID-.DIR_ENTRY( PEID, EID )
    USING STOGROUP -SYSDEFLT-
    PRIQTY 720
    SECQTY 720
    DEFER YES;

CREATE INDEX -DB_USERID-.DIR_ENTRYX2
    ON -DB_USERID-.DIR_ENTRY( EID, DN_TRUNC )
    USING STOGROUP -SYSDEFLT-
    PRIQTY 7200
    SECQTY 3600
    DEFER YES;

CREATE INDEX -DB_USERID-.DIR_ENTRYX3
    ON -DB_USERID-.DIR_ENTRY( DN_TRUNC, EID )
    USING STOGROUP -SYSDEFLT-
    PRIQTY 7200
    SECQTY 3600
    DEFER YES;

-- *********************************
-- Create the DIR_LONGENTRY indexes
-- *********************************
CREATE UNIQUE INDEX -DB_USERID-.DIR_LONGENTRYX1
    ON -DB_USERID-.DIR_LONGENTRY( EID, SEQ )
    USING STOGROUP -SYSDEFLT-
    PRIQTY 720
    SECQTY 720
    CLUSTER
    DEFER YES;

-- *******************************
-- Create the DIR_LONGATTR indexes
-- *******************************
CREATE UNIQUE INDEX -DB_USERID-.DIR_LONGATTRX1
    ON -DB_USERID-.DIR_LONGATTR( EID, ATTR_ID, VALUENUM, SEQ )
    USING STOGROUP -SYSDEFLT-
    PRIQTY 720
    SECQTY 720
    CLUSTER
    DEFER YES;

-- ****************************
-- Create the DIR_CACHE indexes
-- ****************************
CREATE UNIQUE INDEX -DB_USERID-.DIR_CACHEX1
    ON -DB_USERID-.DIR_CACHE( CACHE_NAME, MODIFY_TIMESTAMP )
    USING STOGROUP -SYSDEFLT-
    CLUSTER
    DEFER YES;

-- *****************************
-- Create the DIR_ATTRID indexes
-- *****************************
CREATE UNIQUE INDEX -DB_USERID-.DIR_ATTRIDX1
    ON -DB_USERID-.DIR_ATTRID( ATTR_NOID )
    USING STOGROUP -SYSDEFLT-
    CLUSTER
    DEFER YES;

-- ****************************
-- Create the DIR_DESC indexes
```

```
-- ***************************
CREATE UNIQUE INDEX -DB_USERID-.DIR_DESCX1
    ON -DB_USERID-.DIR_DESC( DEID, AEID )
    USING STOGROUP -SYSDEFLT-
    PRIQTY 7200
    SECQTY 3600
    CLUSTER
    DEFER YES;

-- ****************************
-- Create the DIR_SEARCH indexes
-- ****************************
CREATE INDEX -DB_USERID-.DIR_SEARCHX1
    ON -DB_USERID-.DIR_SEARCH( ATTR_ID, VALUE, EID )
    USING STOGROUP -SYSDEFLT-
    PRIQTY 14400
    SECQTY 7200
    DEFER YES;

CREATE INDEX -DB_USERID-.DIR_SEARCHX2
    ON -DB_USERID-.DIR_SEARCH( EID, ATTR_ID )
    USING STOGROUP -SYSDEFLT-
    PRIQTY 7200
    SECQTY 3600
    CLUSTER
    DEFER YES;

-- ****************************
-- Create the DIR_REPLICA indexes
-- ****************************
CREATE UNIQUE INDEX -DB_USERID-.DIR_REPLICAX1
    ON -DB_USERID-.DIR_REPLICA( REPID )
    USING STOGROUP -SYSDEFLT-
    CLUSTER
    DEFER YES;

-- ****************************
-- Create the DIR_REPENTRY indexes
-- ****************************
CREATE UNIQUE INDEX -DB_USERID-.DIR_REPENTRYX1
    ON -DB_USERID-.DIR_REPENTRY( CHNGID )
    USING STOGROUP -SYSDEFLT-
    CLUSTER
    DEFER YES;

-- ***********************************
-- Create the DIR_LONGREPENTRY indexes
-- ***********************************
CREATE UNIQUE INDEX -DB_USERID-.DIR_LONGREPX1
    ON -DB_USERID-.DIR_LONGREPENTRY( CHNGID , CHNGSEQ )
    USING STOGROUP -SYSDEFLT-
    CLUSTER
    DEFER YES;

-- *************************
-- Create the DIR_EID indexes
-- *************************
CREATE UNIQUE INDEX -DB_USERID-.DIR_EIDX1
    ON -DB_USERID-.DIR_EID( PARTID )
    USING STOGROUP -SYSDEFLT-
    CLUSTER
    DEFER YES;

-- ****************************
-- Create the DIR_REPLSTATUS Index
-- ****************************
CREATE UNIQUE INDEX -DB_USERID-.DIR_REPLSTATUSX1
```

```
    ON -DB_USERID-.DIR_REPLSTATUS( AGREEMENTEID )
    USING STOGROUP -SYSDEFLT-
    CLUSTER
    DEFER YES;

-- *********************************
-- Create the DIR_REPLCHANGE Indexes
-- *********************************
CREATE UNIQUE INDEX -DB_USERID-.DIR_REPLCHANGEX1
    ON -DB_USERID-.DIR_REPLCHANGE( CHANGEID, CONTEXTEID )
    USING STOGROUP -SYSDEFLT-
    CLUSTER
    DEFER YES;

CREATE INDEX -DB_USERID-.DIR_REPLCHANGEX2
    ON -DB_USERID-.DIR_REPLCHANGE( CONTEXTEID )
    USING STOGROUP -SYSDEFLT-
    DEFER YES;

CREATE INDEX -DB_USERID-.DIR_REPLCHANGEX3
    ON -DB_USERID-.DIR_REPLCHANGE( CHANGEID )
    USING STOGROUP -SYSDEFLT-
    DEFER YES;

-- *********************************
-- Create the DIR_LONGREPLCHANGE Index
-- *********************************
CREATE UNIQUE INDEX -DB_USERID-.DIR_LONGREPLCHANGEX1
    ON -DB_USERID-.DIR_LONGREPLCHANGE( CHANGEID, CONTEXTEID,
                                       CHANGESEQ )
    USING STOGROUP -SYSDEFLT-
    CLUSTER
    DEFER YES;

-- *********************************
-- Create the DIR_REPERROR Indexes
-- *********************************
CREATE UNIQUE INDEX -DB_USERID-.DIR_REPLERRORX1
    ON -DB_USERID-.DIR_REPLERROR( ERRORID )
    USING STOGROUP -SYSDEFLT-
    CLUSTER
    DEFER YES;

CREATE INDEX -DB_USERID-.DIR_REPLERRORX2
    ON -DB_USERID-.DIR_REPLERROR( AGREEMENTEID )
    USING STOGROUP -SYSDEFLT-
    DEFER YES;

-- ************************************
-- Create the DIR_LONGREPERROR Indexes
-- ************************************
CREATE UNIQUE INDEX -DB_USERID-.DIR_LONGREPLERRORX1
    ON -DB_USERID-.DIR_LONGREPLERROR( ERRORID, ERRORSEQ )
    USING STOGROUP -SYSDEFLT-
    CLUSTER
    DEFER YES;

CREATE INDEX -DB_USERID-.DIR_LONGREPLERRORX2
    ON -DB_USERID-.DIR_LONGREPLERROR( AGREEMENTEID )
    USING STOGROUP -SYSDEFLT-
    DEFER YES;

-- *********************************
-- Commit all the above SQL statements
-- *********************************
COMMIT;
```

```
-- ********************************************************************
-- Use the following statement if you need to grant EXECUTE privilege
-- on the DB2 CLI plan to the user running the LDAP server.
--
-- NOTE: You need to remove the '--' from each line before you can run
--       these statements.
--
-- Change -DB_PLAN- to the DB2 CLI plan name.
-- Change -LDAP_USERID- to the user ID running the LDAP server.
-- ********************************************************************
--GRANT EXECUTE ON PLAN -DB_PLAN- TO -LDAP_USERID-;
--COMMIT;

-- ********************************************************************
-- Use the following statements if you need to grant SQL SELECT
-- privileges on DB2 catalog tables to the user running the LDAP server.
--
-- NOTE: You need to remove the '--' from each line before you can run
--       these statements.
--
-- Change -LDAP_USERID- to the user ID running the LDAP server.
-- ********************************************************************
--GRANT SELECT ON SYSIBM.SYSCOLUMNS TO -LDAP_USERID-;
--GRANT SELECT ON SYSIBM.SYSCOLDIST TO -LDAP_USERID-;
--GRANT SELECT ON SYSIBM.SYSTABLES TO -LDAP_USERID-;
--GRANT SELECT ON SYSIBM.SYSTABLEPART TO -LDAP_USERID-;
--GRANT SELECT ON SYSIBM.SYSKEYS TO -LDAP_USERID-;
--GRANT SELECT ON SYSIBM.SYSTABLESPACE TO -LDAP_USERID-;
--GRANT SELECT ON SYSIBM.SYSINDEXPART TO -LDAP_USERID-;
--COMMIT;

-- ********************************************************************
-- Use the following statement if you need to grant DBADM privilege
-- on the DB2 database to the user running the LDAP server.
--
-- NOTE: You need to remove the '--' from each line before you can run
--       these statements.
--
-- Change -DB_NAME- to the LDAP database name.
-- Change -LDAP_USERID- to the user ID running the LDAP server.
-- ********************************************************************
--GRANT DBADM ON DATABASE -DB_NAME- TO -LDAP_USERID-;
--COMMIT;
```

The TDBMMGRT SPUFI file

```
-- ********************************************************************
-- *
-- * Licensed Materials - Property of IBM
-- * 5650-ZOS
-- * Copyright IBM Corp.  2006, 2013
-- *
-- ********************************************************************
--
-- Use the SPUFI function to execute these SQL statements.
--
-- Database, table, tablespace and storage class values must be
-- retrieved from your z/OS LDAP Server DB2 configuration.  Use
-- the following DB2 command to query the information:
--
-- '-DISPLAY DATABASE(GLDDB)'
--
-- NOTE: GLDDB is the default database name, your database name may
-- differ.
--
-- ********************************************************************
-- Database Name Information
-- ********************************************************************
-- Change any instance of -DB2_NAME- to the name of the LDAP
-- database name you want to migrate.  Be sure this name matches what
-- was defined for the database name in the z/OS LDAP Server
-- configuration file.
--
-- The default configuration sets this value to GLDDB.  But this
-- value should be verified.
--
-- ********************************************************************
-- DataBase Owner Information
-- ********************************************************************
-- Change any instance of -DB2_USERID- to the database owner id
-- defined as the owner id for the z/OS LDAP Server DB2
-- configuration. This id is the highlevel qualifier for the tables.
--
-- The default configuration sets this value to GLDSRV.  But this
-- value should be verified.
--
-- ********************************************************************
-- LDAP Server User ID Information
-- ********************************************************************
-- Change any instance of -LDAP_USERID- to the user ID for the LDAP
-- server to run under.
--
-- The default configuration sets this value to GLDSRV.  But this
-- value should be verified.
--
-- ********************************************************************
-- Tablespace Information
-- ********************************************************************
-- Change any instance of -MISC_TABLESPACE- to the LDAP miscellaneous
-- tablespace as defined in the z/OS LDAP Server DB2 configuration.
--
-- The default configuration sets this value to MISCTS.  But this
-- value should be verified.
--
-- Change any instance of -REPLICA_TABLESPACE- to the LDAP replica
-- tablespace as defined in the z/OS LDAP Server DB2 configuration.
--
-- The default configuration sets this value to REPTS.  But this
-- value should be verified.
--
-- ********************************************************************
```

```
-- Storage Group Information
-- ****************************************************************
-- Change any instance of -STORAGEGROUP- to the storage group as
-- defined in the z/OS LDAP Server DB2 configuration.
--
-- The default configuration sets this value to SYSDEFLT.  But this
-- value should be verified.

--
--
-- ****************************************************************
-- TABLE OF CONTENTS
-- ****************************************************************
-- SECTION  TITLE
--
-- -------  -------------------------------------------------------
-- 01       SQL statements for migrating from z/OS Integrated
--          Security Services LDAP server to the latest version of
--          IBM Tivoli Directory Server for z/OS LDAP server.
-- -------  -------------------------------------------------------
-- 02       SQL for using the partitioned entry identifier assignment
--          algorithm.  This is generally needed for users migrating
--          from IBM Tivoli Directory Server for z/OS V1R8 to IBM
--          Tivoli Directory Server for z/OS V1R10.
-- -------  -------------------------------------------------------
-- 03       SQL for using advanced replication support.
--          This is generally needed for users migrating
--          from IBM Tivoli Directory Server for z/OS V1R10 to IBM
--          Tivoli Directory Server for z/OS V1R11.
-- -------  -------------------------------------------------------

--
--
-- ****************************************************************
-- *** SECTION 01:                                              ***
-- ***                                                          ***
-- ***    SQL statements for migrating from z/OS Integrated     ***
-- ***    Security Services LDAP server to the latest version of ***
-- ***    IBM Tivoli Directory Server for z/OS LDAP server.      ***
-- ****************************************************************
-- The following SQL statememts, created for the z/OS Integrated
-- Security Services LDAP server, will alter your DB2 database tables
-- such that they will be supported by the latest version of IBM
-- Tivoli Directory Server for z/OS LDAP server.
--
-- ****************************************************************
-- ***** SPECIAL NOTICE *****             ***** SPECIAL NOTICE *****
-- ****************************************************************
-- The DIR_CHANGE table used for replication must be empty before the
-- backend is started on the IBM Tivoli Directory Server for z/OS
-- LDAP server.  The SQL statements in this file will empty the table
-- if it is not empty, but this will result in one or more replicas
-- that are no longer synchronized with this master server.
--
-- To verify that the DIR_CHANGE table is empty, from a separate
-- file, execute the following SQL statement using SPUFI:
--
--    SELECT * FROM -DB2_USERID-.DIR_CHANGE;
--
-- where -DB2_USERID- is changed to the database owner id, as
-- previously described above.
--
-- After execution, if the row count returns 0, then all replicaton
-- changes have been made and the table is ready to be dropped.
--
-- Otherwise, refer to the z/OS Integrated Security Services LDAP
-- Server Administration and Use documentation to ensure all
```

```
-- replication changes have been made and then verify that the table
-- is empty.
--
-- If replication does not take place and the table does not become
-- empty, then all the replication tables will be emptied by the
-- SQL commands below.  In this case replication updates to one or
-- more replicas will have been lost and it will be necessary to
-- synchronize these replicas with the master server. Refer to the
-- IBM Tivoli Directory Server Administration and Use for z/OS
-- publication for information about replica synchronization.
--
-- ********************************************************************
-- Empty the replication tables
-- ********************************************************************
DELETE FROM -DB2_USERID-.DIR_REGISTER;
DELETE FROM -DB2_USERID-.DIR_PROGRESS;
DELETE FROM -DB2_USERID-.DIR_CHANGE;
DELETE FROM -DB2_USERID-.DIR_LONGCHANGE;
--
-- ********************************************************************
-- Drop the replication tables
--
-- NOTE: these SQL statements are commented out.  The IBM Tivoli
-- Directory Server for z/OS LDAP server will ignore these tables.
-- Keeping these tables allows you to roll-back DB2 data to the z/OS
-- Integrated Security Services LDAP server database.  It also allows
-- sharing the database between the two servers.  Remove the '--'
-- characters from the following SQL statements if you want to drop
-- the replication tables.
--
-- NOTE: New replica tables and indexes will be created below.
-- ********************************************************************
-- DROP TABLE -DB2_USERID-.DIR_REGISTER;
-- DROP TABLE -DB2_USERID-.DIR_PROGRESS;
-- DROP TABLE -DB2_USERID-.DIR_CHANGE;
-- DROP TABLE -DB2_USERID-.DIR_LONGCHANGE;
-- COMMIT;

-- ********************************************************************
-- Alter the misc tablespace to set locksize row
-- ********************************************************************
ALTER TABLESPACE -DB2_NAME-.-MISC_TABLESPACE-
        LOCKSIZE ROW;

-- ********************************************************************
-- Alter the misc table to add the schema timestamp column
--
-- NOTE: This SQL statement is commented out.  The IBM Tivoli
-- Directory Server for z/OS LDAP server will automatically alter the
-- DIR_MISC table to add the new schema timestamp column if it is not
-- already there.  As an alternative, you can create the new column
-- here by removing the '--' characters from the following SQL
-- statement.
-- ********************************************************************
-- ALTER TABLE -DB2_USERID-.DIR_MISC
--         ADD SCHEMA_TIMESTAMP TIMESTAMP
--         WITH DEFAULT NULL;

-- ********************************************************************
-- Alter the misc table to add the partitioned eid column
--
-- NOTE: This SQL statement is commented out.  The IBM Tivoli
-- Directory Server for z/OS LDAP server will automatically alter the
-- DIR_MISC table to add the new partitioned eid column if it is not
-- already there.  As an alternative, you can create the new column
-- here by removing the '--' characters from the following SQL
-- statement.
```

```
-- ******************************************************************
-- ALTER TABLE -DB2_USERID-.DIR_MISC
--         ADD PARTITIONED_EID CHAR(1)
--         WITH DEFAULT NULL;

-- ******************************************************************
-- Create the new replica table
-- ******************************************************************
CREATE TABLE -DB2_USERID-.DIR_REPLICA (
        REPID   DECIMAL(15 , 0) NOT NULL,
        CHNGID  DECIMAL(15 , 0) NOT NULL,
        REPCAPS INTEGER         NOT NULL,
        PRIMARY KEY( REPID ) )
IN -DB2_NAME-.-MISC_TABLESPACE-;

-- ******************************************************************
-- Create the new replica entry table
--
-- Note: The maximum size of the change DN column (CHNGDN) in the
--       replication table (DIR_REPENTRY) must be greater than
--       or equal to the maximum size of the DN column in the entry
--       table (DIR_ENTRY).  The default maximum size of 512 is set
--       here.
-- ******************************************************************
CREATE TABLE -DB2_USERID-.DIR_REPENTRY (
        CHNGID    DECIMAL(15 , 0) NOT NULL,
        CHNGFLAGS INTEGER         NOT NULL,
        CHNGSIZE  INTEGER         NOT NULL,
        CHNGDN    VARCHAR(512)    FOR BIT DATA,
        CHNGDATA  LONG VARCHAR    FOR BIT DATA,
        PRIMARY KEY( CHNGID ) )
IN -DB2_NAME-.-REPLICA_TABLESPACE-;

-- ******************************************************************
-- Create the new long replica entry table
-- ******************************************************************
CREATE TABLE -DB2_USERID-.DIR_LONGREPENTRY (
        CHNGID   DECIMAL(15 , 0) NOT NULL,
        CHNGSEQ  INTEGER         NOT NULL,
        CHNGDATA LONG VARCHAR    FOR BIT DATA,
        PRIMARY KEY( CHNGID , CHNGSEQ ) )
IN -DB2_NAME-.-REPLICA_TABLESPACE-;

-- ******************************************************************
-- Create the entry identifier table
-- ******************************************************************
CREATE TABLE -DB2_USERID-.DIR_EID (
        PARTID           DECIMAL(15 , 0) NOT NULL,
        NEXT_EID         DECIMAL(15 , 0) NOT NULL,
        MODIFY_TIMESTAMP TIMESTAMP       NOT NULL,
        PRIMARY KEY( PARTID ) )
IN -DB2_NAME-.-MISC_TABLESPACE-;

-- ******************************************************************
-- Create the new replica index
-- ******************************************************************
CREATE UNIQUE INDEX -DB2_USERID-.DIR_REPLICAX1
        ON -DB2_USERID-.DIR_REPLICA( REPID )
        USING STOGROUP -STORAGEGROUP-
        CLUSTER
        DEFER YES;

-- ******************************************************************
-- Create the new replica entry index
-- ******************************************************************
CREATE UNIQUE INDEX -DB2_USERID-.DIR_REPENTRYX1
        ON -DB2_USERID-.DIR_REPENTRY( CHNGID )
```

```
        USING STOGROUP -STORAGEGROUP-
        CLUSTER
        DEFER YES;

-- *********************************************************************
-- Create the new long replica entry index
-- *********************************************************************
CREATE UNIQUE INDEX -DB2_USERID-.DIR_LONGREPX1
        ON -DB2_USERID-.DIR_LONGREPENTRY( CHNGID , CHNGSEQ )
        USING STOGROUP -STORAGEGROUP-
        CLUSTER
        DEFER YES;

-- *********************************************************************
-- Create the entry identifier index
-- *********************************************************************
CREATE UNIQUE INDEX -DB2_USERID-.DIR_EIDX1
        ON -DB2_USERID-.DIR_EID( PARTID )
        USING STOGROUP -STORAGEGROUP-
        CLUSTER
        DEFER YES;

-- *********************************************************************
-- Update the DB_VERSION value to 4.0 in the DIR_MISC table.
--
-- NOTE: This SQL statement is commented out.  The DB_VERSION value
-- indicates the level of the TDBM database.  Updating the
-- DB_VERSION value to 4.0 enables the backend to use the enhanced
-- sysplex and replication support in the IBM Tivoli Directory Server
-- for z/OS LDAP server.  However, it will prevent the backend
-- database from being shared with a z/OS Integrated Security
-- Services LDAP server.  If you do not plan to share the database,
-- you can update the DB_VERSION value here by removing the '--'
-- characters from the following SQL statement.
-- *********************************************************************
-- UPDATE -DB2_USERID-.DIR_MISC
--      SET DB_VERSION='4.0';

-- *********************************************************************
-- Commit the changes made in this section
-- *********************************************************************
COMMIT;

-- *********************************************************************
-- Grant SQL SELECT privileges on DB2 catalog tables to the user
-- running the LDAP server.
--
-- NOTE: You need to remove the '--' from each line before you can run
--       these statements.
--
-- Change -LDAP_USERID- to the user ID running the LDAP server.
-- *********************************************************************
--GRANT SELECT ON SYSIBM.SYSCOLUMNS TO -LDAP_USERID-;
--GRANT SELECT ON SYSIBM.SYSCOLDIST TO -LDAP_USERID-;
--GRANT SELECT ON SYSIBM.SYSTABLES TO -LDAP_USERID-;
--GRANT SELECT ON SYSIBM.SYSTABLEPART TO -LDAP_USERID-;
--GRANT SELECT ON SYSIBM.SYSKEYS TO -LDAP_USERID-;
--GRANT SELECT ON SYSIBM.SYSTABLESPACE TO -LDAP_USERID-;
--GRANT SELECT ON SYSIBM.SYSINDEXPART TO -LDAP_USERID-;
--COMMIT;

--
--
-- *********************************************************************
-- *** SECTION 02:                                                   ***
-- ***                                                              ***
-- ***  SQL for using the partitioned entry identifier assignment ***
```

```
-- ***  algorithm.  This is generally needed for users migrating  ***
-- ***  from IBM Tivoli Directory Server for z/OS V1R8 to IBM     ***
-- ***  Tivoli Directory Server for z/OS V1R10.                   ***
-- ************************************************************
-- The following SQL statememts, created for the IBM Tivoli Directory
-- Server for z/OS V1R8, will alter your DB2 database tables such
-- that they will support the partitioned entry identifier assignment
-- algorithm.
--
-- You SHOULD NOT run the following SQL statements if your DB2
-- database tables were built to support IBM Tivoli Directory Server
-- for z/OS V1R10 or if your DB2 database tables already support the
-- partitioned entry identifier assignment algorithm.
--
-- If you are unsure whether you need to run the SQL statements in
-- this section, from a separate file, execute the following SQL
-- statement using SPUFI:
--
--    SELECT * FROM SYSIBM.SYSTABLES WHERE
--         CREATOR='-DB2_USERID-' AND NAME='DIR_EID';
--
-- where -DB2_USERID- is changed to the database owner id, as
-- previously described above.
--
-- After execution, if the row count returns 0, then your DB2
-- database tables DO NOT support the partitioned entry identifier
-- assignment algorithm.  If you want to use the partitioned entry
-- identifier assignment algorithm, or if you want to run IBM Tivoli
-- Directory Server for z/OS V1R10, you need to execute the
-- following SQL statements.
--
-- To ensure a successful update of your DB2 database tables, you
-- must comment out all SQL statements in other sections of this file
-- prior to running the following SQL statements.
--
-- ************************************************************
-- Alter the misc tablespace to set locksize row
--
-- NOTE: This SQL statement is commented out.  To set the misc
-- tablespace to locksize row, remove the '--' characters from the
-- following SQL statement.
-- ************************************************************
-- ALTER TABLESPACE -DB2_NAME-.-MISC_TABLESPACE-
--         LOCKSIZE ROW;

-- ************************************************************
-- Alter the misc table to add the partitioned eid column
--
-- NOTE: This SQL statement is commented out.  The IBM Tivoli
-- Directory Server for z/OS LDAP server will automatically alter the
-- DIR_MISC table to add the new partitioned eid column if it is not
-- already there.  As an alternative, you can create the new column
-- here by removing the '--' characters from the following SQL
-- statement.
-- ************************************************************
-- ALTER TABLE -DB2_USERID-.DIR_MISC
--         ADD PARTITIONED_EID CHAR(1)
--         WITH DEFAULT NULL;

-- ************************************************************
-- Create the entry identifier table
--
-- NOTE: This SQL statement is commented out.  To create the entry
-- identifier table, remove the '--' characters from the following
-- SQL statement.
-- ************************************************************
-- CREATE TABLE -DB2_USERID-.DIR_EID (
```

```
--         PARTID           DECIMAL(15 , 0) NOT NULL,
--         NEXT_EID         DECIMAL(15 , 0) NOT NULL,
--         MODIFY_TIMESTAMP TIMESTAMP       NOT NULL,
--         PRIMARY KEY( PARTID ) )
-- IN -DB2_NAME-.-MISC_TABLESPACE-;

-- ********************************************************************
-- Create the entry identifier index
--
-- NOTE: This SQL statement is commented out.  To create the entry
-- identifier index, remove the '--' characters from the following
-- SQL statement.
-- ********************************************************************
-- CREATE UNIQUE INDEX -DB2_USERID-.DIR_EIDX1
--         ON -DB2_USERID-.DIR_EID( PARTID )
--         USING STOGROUP -STORAGEGROUP-
--         CLUSTER
--         DEFER YES;

-- ********************************************************************
-- Commit the changes made in this section
--
-- NOTE: This SQL statement is commented out.  Remove the '--'
-- characters from the following SQL statement to commit the changes
-- made in this section.
-- ********************************************************************
-- COMMIT;

-- ********************************************************************
-- Grant SQL SELECT privileges on DB2 catalog tables to the user
-- running the LDAP server.
--
-- NOTE: You need to remove the '--' from each line before you can run
--       these statements.
--
-- Change -LDAP_USERID- to the user ID running the LDAP server.
-- ********************************************************************
--GRANT SELECT ON SYSIBM.SYSCOLUMNS TO -LDAP_USERID-;
--GRANT SELECT ON SYSIBM.SYSCOLDIST TO -LDAP_USERID-;
--GRANT SELECT ON SYSIBM.SYSTABLES TO -LDAP_USERID-;
--GRANT SELECT ON SYSIBM.SYSTABLEPART TO -LDAP_USERID-;
--GRANT SELECT ON SYSIBM.SYSKEYS TO -LDAP_USERID-;
--GRANT SELECT ON SYSIBM.SYSTABLESPACE TO -LDAP_USERID-;
--GRANT SELECT ON SYSIBM.SYSINDEXPART TO -LDAP_USERID-;
--COMMIT;

--
--
-- ********************************************************************
-- *** SECTION 03:                                                  ***
-- ***                                                             ***
-- ***  SQL for using the advanced replication support.            ***
-- ***  This is generally needed for users migrating               ***
-- ***  from IBM Tivoli Directory Server for z/OS V1R10 to IBM     ***
-- ***  Tivoli Directory Server for z/OS V1R11.                    ***
-- ********************************************************************
-- The following SQL statememts, created for the IBM Tivoli Directory
-- Server for z/OS V1R10, will alter your DB2 database tables such
-- that they will support advanced replication.
--
-- You SHOULD NOT run the following SQL statements if your DB2
-- database tables were built to support IBM Tivoli Directory Server
-- for z/OS V1R11 or if your DB2 database tables already support
-- advanced replication.
--
-- If you are unsure whether you need to run the SQL statements in
-- this section, from a separate file, execute the following SQL
```

```
-- statement using SPUFI:
--
--    SELECT * FROM SYSIBM.SYSTABLES WHERE
--         CREATOR='-DB2_USERID-' AND NAME='DIR_REPLSTATUS';
--
-- where -DB2_USERID- is changed to the database owner id, as
-- previously described above.
--
-- After execution, if the row count returns 0, then your DB2
-- database tables DO NOT support advanced replication.
-- If you want to use advanced replication, you need to execute
-- the following SQL statements.
--
-- To ensure a successful update of your DB2 database tables, you
-- must comment out all SQL statements in other sections of this file
-- prior to running the following SQL statements.
--
-- ********************************************************************
-- Create the DIR_REPLSTATUS table
--
-- NOTE: This SQL statement is commented out.  Remove the '--'
-- characters from the following SQL statement to commit the changes
-- made in this section.
-- ********************************************************************
--CREATE TABLE -DB2_USERID-.DIR_REPLSTATUS (
--    AGREEMENTEID        DECIMAL(15 , 0)   NOT NULL,
--    CHANGEID            INTEGER           NOT NULL,
--    REPLICACAPS         CHAR(64)          FOR BIT DATA,
--    PRIMARY KEY( AGREEMENTEID ) )
--IN -DB2_NAME-.-MISC_TABLESPACE-;

-- ********************************************************************
-- Create the DIR_REPLCHANGE table
--
-- NOTE: This SQL statement is commented out.  Remove the '--'
-- characters from the following SQL statement to commit the changes
-- made in this section.
-- ********************************************************************
--CREATE TABLE -DB2_USERID-.DIR_REPLCHANGE (
--    CHANGEID            INTEGER           NOT NULL,
--    CONTEXTEID          DECIMAL(15 , 0)   NOT NULL,
--    CHANGESIZE          INTEGER           NOT NULL,
--    CHANGEDATA          LONG VARCHAR      FOR BIT DATA,
--    PRIMARY KEY( CHANGEID, CONTEXTEID ) )
--IN -DB2_NAME-.-REPLICA_TABLESPACE-;

-- ********************************************************************
-- Create the DIR_LONGREPLCHANGE table
--
-- NOTE: This SQL statement is commented out.  Remove the '--'
-- characters from the following SQL statement to commit the changes
-- made in this section.
-- ********************************************************************
--CREATE TABLE -DB2_USERID-.DIR_LONGREPLCHANGE (
--    CHANGEID            INTEGER           NOT NULL,
--    CONTEXTEID          DECIMAL(15 , 0)   NOT NULL,
--    CHANGESEQ           INTEGER           NOT NULL,
--    CHANGEDATA          LONG VARCHAR      FOR BIT DATA,
--    PRIMARY KEY( CHANGEID, CONTEXTEID, CHANGESEQ ) )
--IN -DB2_NAME-.-REPLICA_TABLESPACE-;

-- ********************************************************************
-- Create the DIR_REPLERROR table
--
-- NOTE: This SQL statement is commented out.  Remove the '--'
-- characters from the following SQL statement to commit the changes
-- made in this section.
```

```
-- ******************************************************************
--CREATE TABLE -DB2_USERID-.DIR_REPLERROR (
--    ERRORID             INTEGER           NOT NULL,
--    AGREEMENTEID        DECIMAL(15 , 0)   NOT NULL,
--    ERRORSIZE           INTEGER           NOT NULL,
--    ERRORDATA           LONG VARCHAR      FOR BIT DATA,
--    PRIMARY KEY( ERRORID ) )
--IN -DB2_NAME-.-REPLICA_TABLESPACE-;

-- ******************************************************************
-- Create the DIR_LONGREPERROR table
--
-- NOTE: This SQL statement is commented out.  Remove the '--'
-- characters from the following SQL statement to commit the changes
-- made in this section.
-- ******************************************************************
--CREATE TABLE -DB2_USERID-.DIR_LONGREPLERROR (
--    ERRORID             INTEGER           NOT NULL,
--    AGREEMENTEID        DECIMAL(15 , 0)   NOT NULL,
--    ERRORSEQ            INTEGER           NOT NULL,
--    ERRORDATA           LONG VARCHAR      FOR BIT DATA,
--    PRIMARY KEY( ERRORID, ERRORSEQ ) )
--IN -DB2_NAME-.-REPLICA_TABLESPACE-;

-- ******************************************************************
-- Create the DIR_REPLSTATUS Index
--
-- NOTE: This SQL statement is commented out.  Remove the '--'
-- characters from the following SQL statement to commit the changes
-- made in this section.
-- ******************************************************************
--CREATE UNIQUE INDEX -DB2_USERID-.DIR_REPLSTATUSX1
--    ON -DB2_USERID-.DIR_REPLSTATUS( AGREEMENTEID )
--    USING STOGROUP -STORAGEGROUP-
--    CLUSTER
--    DEFER YES;

-- ******************************************************************
-- Create the DIR_REPLCHANGE Indexes
--
-- NOTE: This SQL statement is commented out.  Remove the '--'
-- characters from the following SQL statements to commit the changes
-- made in this section.
-- ******************************************************************
--CREATE UNIQUE INDEX -DB2_USERID-.DIR_REPLCHANGEX1
--    ON -DB2_USERID-.DIR_REPLCHANGE( CHANGEID, CONTEXTEID )
--    USING STOGROUP -STORAGEGROUP-
--    CLUSTER
--    DEFER YES;

--CREATE INDEX -DB2_USERID-.DIR_REPLCHANGEX2
--    ON -DB2_USERID-.DIR_REPLCHANGE( CONTEXTEID )
--    USING STOGROUP -STORAGEGROUP-
--    DEFER YES;

--CREATE INDEX -DB2_USERID-.DIR_REPLCHANGEX3
--    ON -DB2_USERID-.DIR_REPLCHANGE( CHANGEID )
--    USING STOGROUP -STORAGEGROUP-
--    DEFER YES;

-- ******************************************************************
-- Create the DIR_LONGREPLCHANGE Index
--
-- NOTE: This SQL statement is commented out.  Remove the '--'
-- characters from the following SQL statement to commit the changes
-- made in this section.
-- ******************************************************************
```

```
--CREATE UNIQUE INDEX -DB2_USERID-.DIR_LONGREPLCHANGEX1
--    ON -DB2_USERID-.DIR_LONGREPLCHANGE( CHANGEID, CONTEXTEID,
--                                        CHANGESEQ )
--    USING STOGROUP -STORAGEGROUP-
--    CLUSTER
--    DEFER YES;

-- ********************************************************************
-- Create the DIR_REPERROR Indexes
--
-- NOTE: This SQL statement is commented out.  Remove the '--'
-- characters from the following SQL statements to commit the changes
-- made in this section.
-- ********************************************************************
--CREATE UNIQUE INDEX -DB2_USERID-.DIR_REPLERRORX1
--    ON -DB2_USERID-.DIR_REPLERROR( ERRORID )
--    USING STOGROUP -STORAGEGROUP-
--    CLUSTER
--    DEFER YES;

--CREATE INDEX -DB2_USERID-.DIR_REPLERRORX2
--    ON -DB2_USERID-.DIR_REPLERROR( AGREEMENTEID )
--    USING STOGROUP -STORAGEGROUP-
--    DEFER YES;

-- ********************************************************************
-- Create the DIR_LONGREPERROR Indexes
--
-- NOTE: This SQL statement is commented out.  Remove the '--'
-- characters from the following SQL statements to commit the changes
-- made in this section.
-- ********************************************************************
--CREATE UNIQUE INDEX -DB2_USERID-.DIR_LONGREPLERRORX1
--    ON -DB2_USERID-.DIR_LONGREPLERROR( ERRORID, ERRORSEQ )
--    USING STOGROUP -STORAGEGROUP-
--    CLUSTER
--    DEFER YES;

--CREATE INDEX -DB2_USERID-.DIR_LONGREPLERRORX2
--    ON -DB2_USERID-.DIR_LONGREPLERROR( AGREEMENTEID )
--    USING STOGROUP -STORAGEGROUP-
--    DEFER YES;
--
-- ********************************************************************
-- Commit the changes made in this section
--
-- NOTE: This SQL statement is commented out.  Remove the '--'
-- characters from the following SQL statement to commit the changes
-- made in this section.
-- ********************************************************************
-- COMMIT;
```

Appendix C. Supported server controls

The sections that follow describe the supported server controls. For information about ASN.1 (Abstract Syntax Notation One) and BER (Basic Encoding Rules), go to the following website:

`ftp://ftp.rsa.com/pub/pkcs/ascii/layman.asc`

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.

Do Not Replicate

- **Name: Do Not Replicate**
- **Description:** Used by a client to indicate that an add, delete, modify, or modify DN request is not to be replicated to a consumer or forwarding server in an advanced replication environment.
- **Assigned object identifier:** 1.3.18.0.2.10.23
- **Target of control:** Server
- **Control criticality:** Never
- **Values:** There is no value; the **controlValue** field is absent.
- **Detailed description:** This control is valid when sent on a client's add, delete, modify, or modify DN request. The presence of this control indicates that the supplier server in an advanced replication environment should not replicate the update to a consumer or forwarding server.

  Note: The **ldapadd, ldapmodify, ldapmodrdn**, and **ldapdelete** utilities have a **-L** option to add this control to LDAP server requests. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information about the **ldapadd, ldapmodify, ldapmodrdn**, and **ldapdelete** utilities.

IBMLdapProxyControl

- **Name: IBMLdapProxyControl**

- **Description:** Used to provide bind and connection information about extended operation requests that result in LDAP requests to another LDAP server. It is required on **GetDnForUserid** and **GetPrivileges** extended operation requests. The **IBMLdapProxyControl** server control and the **GetDnForUserid** and **GetPrivileges** extended operation requests are deprecated.
- **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
   }
```

- **Detailed description:** This control provides information about extended operation requests that result in the use of the LDAP client to make LDAP requests to another 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 directory 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 directory 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 68 for more information about 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.

If the control is specified more than once in a request, the server returns **LDAP_PROTOCOL_ERROR**.

IBMModifyDNRealignDNAttributesControl

- **Name: IBMModifyDNRealignDNAttributesControl**
- **Description:** Used by a client to request that a Modify DN operation be extended to realign attribute values for attributes with **Distinguished Name** syntax, 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.
- **Detailed description:** This control is valid when sent on a client's Modify DN request. Distinguished names which are renamed might be embedded in DN-syntax attributes throughout the directory contents. You might want 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 attribute types whose syntax is **Distinguished Name** (OID 1.3.6.1.4.1.1466.115.121.1.12), and in the attribute types of **aclEntry** and **entryOwner**, which are known to contain distinguished names. The server evaluates 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 are realigned to match their respective new DN values. If any single access check fails, the entire Modify DN operation fails, and all changes to the directory associated with the current Modify DN operation are undone. The scope for realignment is the backend containing the base DN for the Modify DN request. DN references in other backends or other LDAP servers are not updated. If there are **aclFilter** or **ownerFilter** components in **aclEntry** or **entryOwner** attribute values, the DN references in those filters are not updated.

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 ASN.1 (Abstract Syntax Notation One) syntax describes the BER (Basic Encoding Rules) encoding of the control value.

```
ControlValue ::= SEQUENCE {
   Time Limit INTEGER
}
```

- **Detailed description:** This control is valid when sent on a client's Modify DN request. Modify DN operations might 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), therefore, you might want 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. A time limit of zero causes the control to be ignored. The last time limit value is used if this control is specified more than once.

IBMSchemaReplaceByValueControl

- **Name: IBMSchemaReplaceByValueControl**
- **Description:** Used on a schema modify request to tell the LDAP server that a replace operation replaces all schema values or just matching values.
- **Assigned object identifier:** 1.3.18.0.2.10.20
- **Target of control:** Server
- **Control criticality:** Critical at client's option
- **Values:** The following ASN.1 (Abstract Syntax Notation One) statements describe the BER (Basic Encoding Rules) for encoding the control value using implicit tagging:

```
ControlValue ::= SEQUENCE {
   replaceByValue BOOLEAN
}
```

- **Detailed description:** This control is valid when sent on a client's modify request and has meaning only when performing a modify replace operation of an attribute in the LDAP server schema. If the control value is set to TRUE, then each replace value in the modify operation either replaces the existing value (if there is one with the same object identifier) or is added to the schema (if there is no existing value with the same object identifier). All other values in the schema remain as they are. If the control is set to FALSE, all the values for that attribute in the schema are replaced by the ones specified in the modify operation. See "Updating the schema" on page 297 for more information about how LDAP processes a schema modify with replace operation. In all cases, the values of the attributes that are in the initial LDAP server schema cannot be deleted and can be modified only in very limited ways. See "Updating the schema" on page 297 for more information.

  IBMschemaReplaceByValueControl overrides the **schemaReplaceByValue** server configuration option for the current modify request. The last value is used if this control is specified more than once.

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. The LDAP server does not return a referral even if the requested object is not included in any suffix within the LDAP server and a global referral is defined using the **referral** option in the LDAP server configuration file.

No Replication Conflict Resolution

- **Name: No Replication Conflict Resolution**
- **Description:** Used on an update request from a supplier server to a consumer server in an advanced replication environment to indicate that the consumer server should not resolve any replication conflicts that might occur.
- **Assigned object identifier:** 1.3.18.0.2.10.27
- **Target of control:** Server
- **Control criticality:** Never critical
- **Values:** There is no value; the **controlValue** field is absent.
- **Detailed description:** This control is valid when sent by a supplier server to a consumer server on an add, delete, modify, or modify DN request in an advanced replication environment. The presence of the control indicates that the consumer server does not attempt to resolve any replication conflicts that might occur, such as rejecting an add request because it has an older **createtimestamp** value. In this scenario, the consumer server always accepts the replicated updates and attempts to apply them to the targeted backend.

pagedResults

- **Name: pagedResults**
- **Description:** Used on a SearchRequest and SearchResultDone message to control the rate at which the server returns search results.
- **Assigned object identifier:** 1.2.840.113556.1.4.319
- **Target of control:** Server
- **Control criticality:** Critical at client's option
- **Request values:** The following ASN.1 (Abstract Syntax Notation One) syntax describes the BER (Basic Encoding Rules) encoding of the request control values:

```
RequestControlValue ::= SEQUENCE {
   size         INTEGER,
   cookie       OCTET STRING
}
```

  where,

  – `size` - Specifies the requested page size, the number of entries to return. A page size of zero ends the sequence of paged search requests.
  – `cookie` - Specifies an empty string to obtain the initial page of search results, or specifies the cookie that was returned in the previous paged search response to obtain the next page of search results.

- **Response values:** The following ASN.1 (Abstract Syntax Notation One) syntax describes the BER (Basic Encoding Rules) encoding of the response control values:

```
ResponseControlValue ::= SEQUENCE {
   size         INTEGER,
   cookie       OCTET STRING
}
```

  where,

  – `size` - Specifies the estimated number of entries in the entire result set.
  – `cookie` - Specifies the cookie that is required to retrieve the next page of search results, or specifies an empty string if there are no more entries to return.

- **Detailed description:** Paged search results provide paging capabilities for LDAP client applications that want to receive just a subset of search results instead of the entire result set. The next page of entries is returned to the client application for each subsequent paged search request submitted by the client until the operation is canceled or the last page of results is returned.

  This control is valid when sent on a client's SearchRequest message and when sent back to the client on a SearchResultDone message. Support is provided in the z/OS LDAP client to create and parse the control. See the **ldap_create_page_control()** and **ldap_parse_page_control()** APIs and the **ldapsearch** client utility in *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information.

  See RFC 2696: *LDAP Control Extension for Simple Paged Results Manipulation* for more information about the paged search results control.
- **Server behavior:** By default, the **ibm-slapdPagedResLmt** dynamic configuration attribute in the **cn=configuration** entry is set to 0 which indicates that paged searches are not allowed. Therefore, the **ibm-slapdPagedResLmt** attribute must be set to a nonzero value to allow paged searches. The **ibm-slapdPagedResAllowNonAdmin** and **ibm-slapdPagedResLmt** dynamic configuration attributes in the **cn=configuration** entry and the **idleConnectionTimeout** configuration option can be used to limit server resources used by paged searches. The **PersistentSearch** and **pagedResults** server controls cannot both be specified on a search request. See "cn=configuration" on page 139 for more information about the dynamic configuration attributes in the **cn=configuration** entry. The server ignores the paged search request control if the page is greater than or equal to the size limit value specified in the search request. A paged search response control is not returned by the server in this case.

PasswordPolicy

- **Name: PasswordPolicy**
- **Description:** Used by client applications on add, bind, compare, and modify requests to obtain additional warning or error information about a user's password value.
- **Assigned object identifier:** 1.3.6.1.4.1.42.2.27.8.5.1
- **Target of control:** Server
- **Control criticality:** Never critical
- **Request values:** There is no value; the **controlValue** field is absent.
- **Response values:** The following ASN.1 (Abstract Syntax Notation One) syntax describes the BER (Basic Encoding Rules) encoding of the control value.

```
ControlValue ::= SEQUENCE {
   warning [0] CHOICE OPTIONAL {
      timeBeforeExpiration  [0] INTEGER (0 .. maxInt),
      graceLoginsRemaining  [1] INTEGER (0 .. maxInt) }
   error   [1] ENUMERATED OPTIONAL   {
      passwordExpired              (0),
      accountLocked                (1),
      changeAfterReset             (2),
      passwordModNotAllowed        (3),
      mustSupplyOldPassword        (4),
      insufficientPasswordQuality  (5),
      passwordTooShort             (6),
      passwordTooYoung             (7),
      passwordInHistory            (8) }
}
```

Where,

– `warning` - An optional field that indicates the password policy warning code. If `timeBeforeExpiration` is set, the integer indicates the number of seconds before the bound user's password expires. If `graceLoginsRemaining` is specified, it indicates the remaining number of log ins the bound user has before the password expires.

– `error` - An optional field that indicates the password policy error code.

- **Detailed description:** This control is valid when sent on an LDAP client's add, bind, compare, or modify request to the LDAP server. The LDAP server returns the **PasswordPolicy** response control to the client that contains additional warning and error information about a user's password value. For example, on bind and compare requests, the LDAP server may send a **PasswordPolicy** response control to the client that indicates that the bound user's password is about to expire, has expired, or must be changed after being reset by an LDAP administrator. While on add and modify requests of password values, the LDAP server may send a **PasswordPolicy** response control that indicates the password is too short, does not meet password policy quality standards, or the password value exists in the password history of the entry being modified. This information is sent to the client on the add, bind, compare, or modify response.

  Note:

  1. The LDAP server does not send a **PasswordPolicy** response control when a Kerberos (GSSAPI) or EXTERNAL bind is done.
  2. The LDAP client utilities automatically send the **PasswordPolicy** control as a noncritical control on add, bind, compare, and modify requests to the targeted LDAP server. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information.

PersistentSearch

- **Name: PersistentSearch**
- **Description:** 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
}

EntryChangeNotification ::= SEQUENCE {
   changeType ENUMERATED {
            add (1),
            delete (2),
            modify (4),
            moddn (8) },
   previousDN LDAPDN OPTIONAL,
   changeNumber INTEGER OPTIONAL
}
```

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 modify DN changes.
- `changesOnly` - A flag that, if **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` - A flag that, if **TRUE**, an **entryChangeNotification** control is included when returning a changed entry that matches the search. If set to **FALSE**, the control is not included.
- `changeType` - Indicates the type of change made to the entry.
- `previousDN` - For a moddn changeType, the DN of the entry before it was renamed.
- `changeNumber` - The changeNumber of the change log entry, if any, that was created for this change.

- **Detailed description:** The control is valid when sent on a client's 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 IBM Tivoli Directory Server Client Programming for z/OS* 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 supported in the TDBM, CDBM, LDBM, and GDBM backends. In addition, the schema entry (cn=schema) and the rootDSE (zero-length DN) support persistent searches. 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 83 for more information about 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 dereference option was set to something other than **LDAP_DEREF_NEVER** or **LDAP_DEREF_FINDING**, **LDAP_PROTOCOL_ERROR** is returned. If **LDAP_DEREF_FINDING** is specified, alias dereferencing is performed when the persistent search is issued to determine the real base entry. The dereferenced base entry is then used to determine if modified entries are within the scope of the persistent search request.
  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 afterward. No search results are returned for entries moved out of the search filter or scope because of modification or rename.
8. For a persistent search of the root DSE, the search scope must be **LDAP_SCOPE_SUBTREE**. Backends that do not support persistent search or do not have persistent search enabled is skipped if a null-based subtree search is used and the persistent search control is marked as critical, otherwise a typical search is performed for those backends.
9. If a **PersistentSearch** control is included in a search request for a TDBM, CDBM, LDBM, 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 'typical' search is performed (even if changesOnly is **TRUE**).
10. Change log entries trimmed by the LDAP server because of 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 such as typical entries, even if the **manageDsaIT** control is not specified on the persistent search.
12. Idle connection timeout also affects persistent search connections. See the description of the **idleConnectionTimeout** configuration option in Chapter 8, "Customizing the LDAP server configuration," on page 83 for more information.
13. **sizeLimit** and **timeLimit** parameters and configuration options and group search limits 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, timeLimit**, and groups search limits are ignored.
14. Only the entry specified in a modify DN 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 sysplex environment, only one persistent search request must be made to one of the LDAP servers in the sysplex. The sysplex support results in all the LDAP servers participating in the persistent search. The exception to this is if the target of the search is a TDBM backend where the server compatibility level is less than 4. In this case, an identical persistent search request must be issued to each LDAP server in the sysplex. See the description of the **serverCompatLevel** configuration option in "serverCompatLevel {3 | 4 | 5 | 6 | 7| 8}" on page 125 for more information about the server compatibility level.

    Persistent search against a TDBM backend in a sysplex environment with large groups can result in performance problems. See "Large static groups considerations" on page 622 for more information.

16. The SDBM backend does not support persistent search. To be notified of changes to a RACF user (including password changes), group, user-group connection, or general resource profile, 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 user, group, connection, or resource profile.
17. Operational attributes are returned on persistent searches except the following: **aclSource, hasSubordinates, ibm-allGroups, ibm-allMembers, ibm-entryChecksum, ibm-entryChecksumOp, ibm-replicationChangeLdif, ibm-replicationFailedChangeCount, ibm-replicationFailedChanges, ibm-replicationIsQuiesced, ibm-replicationLastActivationTime, ibm-replicationLastChangeId, ibm-replicationLastFinishTime, ibm-replicationLastResult, ibm-replicationLastResultAdditional, ibm-replicationNextTime, ibm-replicationPendingChangeCount, ibm-replicationPendingChanges, ibm-replicationPerformance, ibm-replicationState, ibm-replicationThisServerIsMaster, ownerSource**, and **pwdChangedTime**. The **aclEntry, aclPropagate, entryOwner**, and **ownerPropagate** attributes are returned only if they are defined for the entry and are not inherited from a superior entry.

Refresh Entry

- **Name: Refresh Entry**
- **Description:** Used by a consumer server in an advanced replication environment to notify a supplier server that a replication conflict has occurred during a modify request.
- **Assigned object identifier:** 1.3.18.0.2.10.24
- **Target of control:** Server
- **Control criticality:** Never critical
- **Values:** There is no value; the **controlValue** field is absent.
- **Detailed description:** This control is valid when sent on an LDAP modify response from a consumer to a supplier in an advanced replication environment when a replication conflict is detected in an entry on a consumer server. When the supplier server receives this control along with an **LDAP_OTHER** return code from the consumer server, the supplier sends its copy of the modified entry to the consumer with the intention of bringing the consumer server back in sync.

replicateOperationalAttributes

- **Name: replicateOperationalAttributes**
- **Description:** Used to pass the values of operational attributes that are typically set by the server during an add, modify, or modify DN operation.
- **Assigned object identifier:** 1.3.18.0.2.10.19
- **Target of control:** Server
- **Control criticality:** Critical at client's option
- **Values:** The values in this control identify the operational attributes and values to be set. The following ASN.1 (Abstract Syntax Notation One) syntax describes the BER (Basic Encoding Rules) encoding of the control value.

```
ControlValue ::= SEQUENCE OF SEQUENCE {
  operation  ENUMERATED {
                add     (0),
                delete  (1),
                replace (2) },
  modification    AttributeTypeAndValues
}
```

```
AttributeTypeAndValues ::= SEQUENCE {
  type OCTET STRING,
  vals SET OF OCTET STRING
}
```

where:

- `operation` - Indicates whether the operational attribute value should be added to the entry, should be deleted from the entry, or should replace the current value in the entry.
- `type` - Specifies the name of the operational attribute.
- `vals` - Specifies the values of the operational attribute.

- **Detailed description:** This control is intended to be used to pass values to the server for operational attributes that are typically set by the server, not by the client. For example, a master server might use this control to pass the **modifiersName** and **modifyTimestamp** values on a replication request because the entry on the replica has the same values as on the master.
- **Server behavior:**
  1. The control is only supported on an add, modify, or modify DN request on a basic replication peer or read-only replica server or an advanced replication consumer server. If the control is specified on another request and the control is critical, the server returns **LDAP_UNAVAILABLE_CRITICAL_EXTENSION**.
  2. If using basic replication, the requester must be bound as the master server DN or peer server DN for the backend processing the request, as specified by the **masterServerDN** or **peerServerDN** configuration option in the backend section of the LDAP server configuration file. If using advanced replication, the requester must be bound as the DN specified as the **ibm-replicaCredentialsDN** attribute value in the replication agreement. If the requester is not bound in any of these manners and the control is critical, the server returns **LDAP_UNAVAILABLE_CRITICAL_EXTENSION**.
  3. For an add request and for a modify DN request of a TDBM entry, the `operation` specified in the control value cannot be `delete`. If `delete` is specified, the server returns **LDAP_UNWILLING_TO_PERFORM**. `delete` is supported for a modify request and for a modify DN request of an LDBM or CDBM entry.
  4. Each attribute type specified in the control must be defined in the LDAP server schema. If it is not, the server returns **LDAP_UNDEFINED_TYPE** if the control is critical, otherwise it ignores the attribute.
  5. There is no ACL checking performed for the changes to the entry resulting from the control. The server does perform schema checking to assure the attributes are allowed in the entry.
  6. If more than one **replicateOperationalAttributes** control is specified in a request, the server returns **LDAP_PROTOCOL_ERROR**.

Replication bind failure time stamp control

- **Name: Replication bind failure time stamp control**
- **Description:** Used to propagate bind failure time stamp values for user entry password policy attributes between a master server and read-only replica server consistently.
- **Assigned object identifier:** 1.3.18.0.2.10.34
- **Target of control:** Server

- **Control criticality:** Never critical
- **Values:** Time stamp of the bind operation in string format.
- The value is absent on the request control. The value is a time stamp in string format for the response control. This is the time stamp of the bind operation that is used in updating password policy operational attributes on the master server.
- **Detailed description:** This control is valid on a bind request that uses simple authentication. The control is used in an advanced replication environment to manage **pwdFailureTime, pwdGraceUseTime**, and **pwdExpirationWarned** consistently between a read-only replica server and a master server. Any authentication request to a read-only replica that updates password policy operational attributes in the user entry includes this control on a chained bind request to the master server. This triggers a similar update on the master during the chained bind. The bind response includes this control, and when appropriate, a time stamp value is returned representing the time stamp used on the master server in any of the appropriate operational attributes. The read-only replica server then uses the returned time stamp to ensure that the attributes are managed consistently during the operation and subsequent replication of attributes from the master server to the replica server. Use of this control requires that the 'replication of bind failure on read-only replica' feature is enabled on all servers in the advanced replication topology.

Replication Supplier ID Bind

- **Name: Replication Supplier ID Bind**
- **Description:** Used by supplier gateway server when it binds to a consumer server in an advanced replication environment.
- **Assigned object identifier:** 1.3.18.0.2.10.18
- **Target of control:** Server
- **Control criticality:** Never critical
- **Values:** The following ASN.1 (Abstract Syntax Notation One) syntax describes the BER (Basic Encoding Rules) encoding of the control value.

```
ControlValue ::= SEQUENCE {
  SupplierServerId OCTET STRING
}
```

  where,

  `SupplierServerId` - A string containing the advanced replication supplier server id.
- **Detailed description:** This control is used by a gateway server to determine which servers to replicate to. Gateway servers only replicate updates that are received from other gateway servers to their own local servers (servers that exist in the same site as the gateway server, including peer and forwarding servers). When a gateway server binds to its consumer servers, this control is sent with its own server ID as the control value. When a gateway server receives such a control in a bind request, it knows that a gateway server is bound as a supplier and that only local servers should receive replicated updates.
- **Server behavior:** This control is only sent by a gateway server in an advanced replication environment when bound as the master server distinguished name specified in the replication agreement entry. If this control is sent by a user who does not have access, an **LDAP_UNWILLING_TO_PERFORM** error is returned.

Server Administration

- **Name: Server Administration**

- **Description:** Used by an LDAP administrator on an add, delete, modify, or modify DN operation under conditions where the operation is typically refused. Administrative role permissions are still enforced.
- **Assigned object identifier:** 1.3.18.0.2.10.15
- **Target of control:** Server
- **Control criticality:** Critical at client's option
- **Values:** There is no value; the **controlValue** field is absent.
- **Detailed description:** In an advanced replication environment, this control allows an add, delete, modify, or modify DN operation sent by an LDAP administrator to be processed by a server that typically refuses the operation, such as a quiesced forwarding server or a read-only replica server. The processed operation is then replicated as any other update. This control is used by the **ldapdiff** utility to enable updates to occur on consumer servers that are no longer synchronized with the supplier server.

  Note: This control must be used with discretion because entry updates are allowed under unusual circumstances. Therefore, it is the user's responsibility to ensure the server being updated ends up in a state consistent with the other servers in an advanced replication environment. For example, in an advanced replication environment, the entry's **modifyTimestamp** attribute value, which is used as the base for conflict resolution, might be different on different servers if the entry gets updated individually on those servers with this control.
- **Server behavior:** This control can only be specified by a user bound as an LDAP administrator. If user is not bound as an LDAP administrator, the server returns an **LDAP_UNWILLING_TO_PERFORM** error. If the server is a supplier or consumer server and is quiesced in an advanced replication environment, the control must be specified in order to allow the update to occur.

  Note: The **ldapadd, ldapmodify, ldapmodrdn**, and **ldapdelete** utilities have a **-k** option to add this control to LDAP server requests. See *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information about the **ldapadd, ldapmodify, ldapmodrdn**, and **ldapdelete** utilities.

SortKeyRequest

- **Name: SortKeyRequest**
- **Description:** Used on a SearchRequest message to specify the criteria that a server should use to sort the results of an LDAP search request.
- **Assigned object identifier:** 1.2.840.113556.1.4.473
- **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 OF SEQUENCE {
   attributeType           AttributeDescription,
   orderingRule      [0]   MatchingRuleId OPTIONAL,
   reverseOrder      [1]   BOOLEAN DEFAULT FALSE
}
```

  where,

  – `attributeType` - Specifies an attribute name that the server should use to sort the search results.

– `orderingRule` - An optional field that specifies the name or OID of a matching rule that the server should use when sorting by `attributeType`.
– `reverseOrder` - An optional boolean field that determines whether to do the sort in reverse order.

- **Detailed description:** Sorted search results provide sort capabilities for LDAP client applications that have limited or no sort functionality. Sorted search results allow z/OS LDAP client applications to receive search results sorted based on a list of criteria, where each criterion is a sort key that includes an attribute type and optional matching rule and descending order. The server uses the criteria to sort search results before returning them.

  The **SortKeyRequest** server control is valid when sent on a client's SearchRequest message. Support is provided in the z/OS LDAP client to create the sort list and the control. See the **ldap_create_sort_keylist(), ldap_create_sort_control()**, and **ldap_free_sort_keylist()** APIs and the **ldapsearch** client utility in *z/OS IBM Tivoli Directory Server Client Programming for z/OS* for more information.

  See RFC 2891: *LDAP Control Extension for Server Side Sorting of Search Results* and "SortKeyResponse" on page 693 for more information about the sorted search controls.
- **Server behavior:** By default, the **ibm-slapdSortKeyLimit** dynamic configuration attribute in the **cn=configuration** entry is set to 0 which indicates that sorted searches are not allowed. Therefore, the **ibm-slapdSortKeyLimit** attribute must be set to a nonzero value to allow sorted searches. The **ibm-slapdSortSrchAllowNonAdmin** and **ibm-slapdSortKeyLimit** dynamic configuration attributes in the **cn=configuration** entry can be used to limit server resources used by sorted searches. The **PersistentSearch** and **SortKeyRequest** server controls cannot both be specified on a search request. See "cn=configuration" on page 139 for more information about the dynamic configuration attributes in the **cn=configuration** entry.

  The following describes the server handling of the sort keys:

  1. A NULL attribute value is always treated as being a larger value than all other valid values when sorting.
  2. An entry that matches the search criteria but does not contain a matching sort key attribute is sorted as if it has a single NULL value.
  3. If an entry matches the search criteria but the bound user does not have read access to a matching sort key attribute, the attribute is treated as if it has a single NULL value.
  4. With the exception of SDBM entries, sorting can even be performed on attribute values in the entry which are not requested in the return data. An example would be where the search request sorts on the **sn** attribute, but only specifies that the **cn** and **objectclass** attributes be returned. Another example is when the search request specifies that only attribute types and not values are to be returned. For SDBM entries returned on a search request, sorting is only performed on attribute values included in the returned data.
  5. If a sort is performed on a multi-valued attribute that contains multiple values, the sort is performed on the least value.
  6. The **ibm-slapdDN** attribute is specified in a sort key to sort search results by entry DN.
  7. If an `orderingRule` value is specified in a sort key, the server checks that the value identifies an ordering rule that is valid for the syntax of the attribute specified in the sort key, but the server might not use the rule. Instead, the server always uses the ordering rule associated with the definition of the

attribute in the schema. See Chapter 15, “LDAP directory schema,” on page 275 for more information about ordering rules.

SortKeyResponse

- **Name: SortKeyResponse**
- **Description:** Used on a SearchResultDone message to return the result of a sorted search.
- **Assigned object identifier:** 1.2.840.113556.1.4.474
- **Target of control:** Client
- **Control criticality:** Never critical
- **Values:** The following ASN.1 (Abstract Syntax Notation One) syntax describes the BER (Basic Encoding Rules) encoding of the control value:

```
ControlValue ::= SEQUENCE {
  sortResult ENUMERATED {
   success                      (0),   -- results are sorted
   operationsError              (1),   -- server internal failure
   timeLimitExceeded            (3),   -- timelimit reached
                                          before sorting was completed
   adminLimitExceeded          (11),   -- too many matching entries
                                          for the server to sort
   noSuchAttribute             (16),   -- unrecognized attribute type
                                          in sort key
   inappropriateMatching       (18),   -- unrecognized or inappropriate
                                          matching rule in sort key
   insufficientAccessRights    (50),   -- refused to return sorted results
                                          to this client
   busy                        (51),   -- too busy to process
   unwillingToPerform          (53),   -- unable to sort
   other                       (80)
  },
  attributeType[0]         AttributeDescription         OPTIONAL
}
```

  where,

  – `sortResult` - Specifies the sort result return code.

  – `attributeType` - An optional field that specifies the returned attribute name associated with a sort error.

- **Detailed description:** This response control allows a client to determine the result of a sorted search request. The control is included in the server's SearchResultDone message. Support is provided in the z/OS LDAP client to parse this control. See the **ldap_parse_sort_control()** API and the **ldapsearch** client utility in the *z/OS IBM Tivoli Directory Server Client Programming for z/OS* and “SortKeyRequest” on page 691 for more information.

  See RFC 2891: *LDAP Control Extension for Server Side Sorting of Search Results* for more information about the sorted search controls.

- **Server behavior:** See the server behavior for “SortKeyRequest” on page 691.

Appendix D. Supported extended operations

The sections that follow describe the supported extended operations. For information about ASN.1 (Abstract Syntax Notation One) and BER (Basic Encoding Rules), see:

`ftp://ftp.rsa.com/pub/pkcs/ascii/layman.asc`

Account status

- **Name: Account status**
- **Description:** Used to query the status of a user entry that contains a **userPassword** value. The status returned is whether the user's account is opened, locked by an administrator, or the password is expired.
- **Assigned object identifier:** 1.3.18.0.2.12.58
- **Values:** The following ASN.1 syntax describes the BER encoding of the request value.

```
RequestValue ::= SEQUENCE  {
 userDN LDAPDN
}
```

  where,

  `userDN` - A distinguished name (DN) containing the entry whose account status is being queried.
- **Detailed description:** The **Account status** extended operation is only allowed when bound as an LDAP root or directory administrator, or as a user querying their own account status.
- **Response object identifier:** 1.3.18.0.2.12.59
- **Response description:** This response is used to return whether the specified user is able to authenticate to the server (open), the password is expired, or the account is locked.
- **Response values:** The following describes the response value.

```
ResponseValue ::= SEQUENCE  {
 status  ENUMERATED {
    open         (0),
    locked       (1),
    expired      (2)  }
}
```

- **Response detailed description:**

  The following table summarizes some different error scenarios and the **Account status** response returned for such scenarios.

Error scenario	Account status response
An unauthorized user tries to perform the extended operation	Returns an **LDAP_INSUFFICIENT ACCESS** return code
Syntax of DN specified is not correct	Returns an **LDAP_INVALID_DN_SYNTAX** return code
No results returned	Returns an **LDAP_NO_RESULTS_RETURNED** return code

Error scenario	Account status response
`userDN` does not exist	Returns an **LDAP_NO_SUCH_OBJECT** return code
Internal server error	Returns an **LDAP_OPERATIONS_ERROR** return code
LDAP server is unable to decode the request	Returns an **LDAP_PROTOCOL_ERROR** return code
Returned for errors not covered by previously documented return codes. Check the corresponding error message for further details.	Returns an **LDAP_OTHER** return code

Cascading control replication

- **Name: Cascading control replication**
- **Description:** Performs the requested action to the specified server and passes it along to all replicas of the given replication context. If any of these are forwarding replicas or gateway servers, they pass the extended operation along to their replicas. The operation cascades over the entire advanced replication topology. This extended operation should be targeted against a master server.
- **Assigned object identifier:** 1.3.18.0.2.12.15
- **Values:** The following ASN.1 syntax describes the BER encoding of the request value.

```
RequestValue ::= SEQUENCE  {
 action INTEGER {
    quiesce            (0),
    unquiesce          (1),
    replicateNow       (2),
    wait               (3) },
 contextDN LDAPDN,
 timeout INTEGER
}
```

where,

`action` - An integer value indicating the operation to be performed on the specified server.

– If set to **quiesce**, updates under the replication context `contextDN` are restricted to LDAP administrators with the appropriate authority, if using the **Server Administration** control (OID 1.3.18.0.2.10.15), and any replication master server DNs with authority under this context. Advanced replication continues for a quiesced context. If the server is restarted, all replication contexts are then unquiesced.

– If set to **unquiesce**, updates under the replication context `contextDN` are allowed and normal operation resumes.

– If set to **replicateNow**, all queued updates for each replication agreement under `contextDN` are immediately replicated, regardless of schedule. After queued replication updates have been replicated, each replication agreement follows its normal schedule. If there are any suspended replication agreements, they are skipped and any queued updates remain queued for those replication agreements. Unlike **wait**, this extended operation is propagated to the consumer server of each replication agreement without waiting for all queued updates to be applied.

– If set to **wait**, all queued updates for each replication agreement under `contextDN` are immediately replicated, regardless of schedule. After queued replication updates have been replicated, each replication agreement follows its normal schedule. If there are any suspended replication agreements, they are skipped and any queued updates remain queued for those replication agreements. Unlike **replicateNow**, this extended operation is not propagated to the consumer server of each replication agreement until that agreement is finished replicating.

`contextDN` - A distinguished name (DN) containing the replication context that this operation affects.

`timeout` - An integer value indicating the number of seconds that the extended operation must successfully complete. If not present, or 0, the operation has an indefinite amount of time to complete.

- **Detailed description:** The **Cascading control replication** extended operation returns when one of the following conditions occurs:
  – The request is complete on all servers.
  – A failure has occurred on one of the servers in the replication topology.
  – The `timeout` value is exceeded.

  The **Cascading control replication** extended operation is allowed only when the bound user has update authority to all replication agreements in the specified `contextDN` or is authenticated as a master server for the specified `contextDN`.
- **Response object identifier:** 1.3.18.0.2.12.15
- **Response description:** This extended operation response is used to return error information when a problem is encountered with performing the **Cascading control replication** extended operation.
- **Response values:** The following describes the response value.

```
ResponseValue ::= SEQUENCE {
   resultCode                 INTEGER,
   msg                        OCTET STRING,
   supplier                   OCTET STRING,
   consumer                   OCTET STRING,
   additionalResultCode[1]    INTEGER OPTIONAL {
       LDAP_REPLICATION_SUCCESS (0),
       LDAP_REPLICATION_RETRYING (2) },
   agreementDN[2]             LDAPDN OPTIONAL
}
```

where,

`resultCode` - An integer value indicating whether the extended operation is successful. Standard LDAP return codes are returned as values within this portion of the extended operation response.

`msg` - A string containing a reason code message. In most cases, it is an error message indicating why the extended operation failed. In some cases, it can be an informational or warning message.

`supplier` - A string containing the shortened host name and advanced replication server ID of the supplier server targeted by the extended operation. The format is '*shortenedHostName*:*serverID*'. If the shortened host name cannot be determined, the format is 'server ID:*serverID*'.

`consumer` - A string containing the *host*:*port* of the consumer server that is reporting an error. This is only returned when the consumer has a problem performing the requested operation.

`additionalResultCode` - An integer value that is returned when the `resultCode` is set to **LDAP_TIMEOUT**.

agreementDN - A distinguished name (DN) containing the replication agreement that is in error.

- **Response detailed description:**

  The following table summarizes some different error scenarios and the **Cascading control replication response** returned for such scenarios.

Error scenario	Cascading control replication response
contextDN does not exist or is not a replication context	Returns an **LDAP_NO_SUCH_OBJECT** return code
Backend does not support advanced replication	Returns an **LDAP_UNWILLING_TO_PERFORM** return code
Not authorized to perform operation	Returns an **LDAP_INSUFFICIENT_ACCESS** return code
Operation did not complete within the specified time	Returns an **LDAP_TIMEOUT** return code
Syntax of DN specified is not correct	Returns an **LDAP_INVALID_DN_SYNTAX** return code
Value for the input option is not valid	Returns an **LDAP_PROTOCOL_ERROR** return code
LDAP server is unable to decode the request	Returns an **LDAP_DECODING_ERROR** return code

When a **Cascading control replication response** returns an **LDAP_TIMEOUT** return code, the additionalResultCode field in the **Cascading control replication response** is set to either 0 or **LDAP_REPLICATION_RETRYING** to indicate the replication update is being retried. The **LDAP_REPLICATION_RETRYING** error is only returned when action is set to **wait**.

changeLogAddEntry

- **Name: changeLogAddEntryRequest**
- **Description:** Causes the LDAP server to create a change log entry in the change log by 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,
 class                        OCTET STRING,
 resource                     OCTET STRING,
 changeType                   INTEGER {
    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 2. If the interface is extended in the future then other values are supported.

`applicationID` - 1 for RACF. Other applications have different identifiers. The identifier informs the LDAP server which (if any) translations of the data should be done.

`userid` - A string containing the user ID that is 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 is created, modified, deleted, or renamed. The RACF application can specify a value for both user ID 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.

`class` - A string containing the class of the resource profile that is created, modified, deleted, or renamed. This string is used along with the resource string to form a resource profile DN as the value of the **targetDN** attribute in the change log entry.

`resource` - A string containing the resource profile that is created, modified, deleted, or renamed. This string is used along with the class string to form a resource profile DN as the value of the **targetDN** attribute in the change log entry.

`changeType` - An integer 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`.`*uuuuuu*`Z`

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 Coordinated Universal Time.

`initiator` - A string containing the user ID 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    SEQUENCE OF AttributeValue,
 action  ENUMERATED {
      add (0),
      replace (1),
      delete (2) },
 requestValue Boolean }
```

Where,

`field` - 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` - 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` - 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.

- **Detailed description:** Class and resource cannot be specified with user ID or group. Both class and resource must be specified if either one is specified. In this case, SDBM must be configured to support RACF resources, by specifying **enableResources on** in the SDBM section of the LDAP server configuration file.
- **Response object identifier:** 1.3.18.0.2.12.49
- **Response description:** This response is used to return error information if an incorrect **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)
  }
  msg         OCTET STRING
 }
```

- **Response detailed description:**

  The following table summarizes some different error scenarios and the **changeLogAddEntryRequest** response returned for such scenarios.

Error scenario	changeLogAddEntryRequests 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 user ID to an LDAP DN	Returns a **dnConvertFailed** return code

Control replication

- **Name: Control replication**
- **Description:** Used to suspend replication, resume replication, or force immediate replication by a supplier server in an advanced replication environment. When a replication agreement is suspended, updates under the context are allowed but

the agreement queues the updates to its replica server until advanced replication is resumed for the agreement. This extended operation should be targeted against a master server.

- **Assigned object identifier:** 1.3.18.0.2.12.16
- **Values:** The following ASN.1 syntax describes the BER encoding of the request value.

```
RequestValue ::= SEQUENCE {
 action INTEGER {
     suspend            (0),
     resume             (1),
     replicateNow       (2) },
 scope INTEGER {
     singleAgreement    (0),
     allAgreements      (1) },
 entryDN LDAPDN
}
```

Where,

`action` - An integer value indicating the operation to be performed on the supplier server. If set to **suspend**, the replication agreement queues the updates to its replica server until advanced replication is resumed for the agreement. If set to **resume**, advanced replication for the replication agreement continues. If set to **replicateNow** and the replication agreement is waiting for scheduled replication to occur, any outstanding updates are immediately replicated. **replicateNow** has no effect on a suspended replication agreement.

`scope` - An integer value indicating the extent of the action that is to be performed. If set to **singleAgreement**, the request applies to a single replication agreement. If set to **allAgreements**, the request applies to all replication agreements within a replication context. This parameter indicates whether the `entryDN` is a replication agreement or context entry.

`entryDN` - A distinguished name (DN) containing the replication context or agreement that this operation affects. If `scope` is set to **singleAgreement**, this specifies the distinguished name of the replication agreement that this extended operation is acting on. If `scope` is set to **allAgreements**, this specifies the distinguished name of the replication context and indicates that all agreements within the context are to be acted on.

- **Response object identifier:** 1.3.18.0.2.12.16
- **Response description:** This extended operation response is used to return error information when a problem is encountered with performing the **Control replication** extended operation.
- **Response values:** The following ASN.1 syntax describes the BER encoding of the response value.

```
ResponseValue :: = SEQUENCE {
 resultCode          INTEGER,
 msg                 OCTET STRING,
 consumer            OCTET STRING
}
```

Where,

`resultCode` - An integer value indicating whether the extended operation is successful. Standard LDAP return codes are returned as values within this portion of the extended operation response.

`msg` - A string containing a reason code message. In most cases, it is an error message indicating why the extended operation failed. In some cases, it can be an informational or warning message.

consumer - A string containing the *host*:*port* of the consumer server that is reporting an error. This is returned when the consumer has certain problems performing the requested operation.

- **Response detailed description:**

  The following table summarizes some different error scenarios and the **Control replication response** returned for such scenarios.

Error scenario	Control replication response
entryDN does not exist or is not a replication context or agreement	Returns an **LDAP_NO_SUCH_OBJECT** return code
Backend does not support advanced replication	Returns an **LDAP_UNWILLING_TO_PERFORM** return code
Not authorized to perform operation	Returns an **LDAP_INSUFFICIENT_ACCESS** return code
Syntax of DN specified is not correct	Returns an **LDAP_INVALID_DN_SYNTAX** return code
Value for the input option is not valid	Returns an **LDAP_PROTOCOL_ERROR** return code
LDAP server is unable to decode the request	Returns an **LDAP_DECODING_ERROR** return code

Control replication error log

- **Name: Control replication error log**
- **Description:** Used to display advanced replication errors in the error log and correct any advanced replication problems that occur.
- **Assigned object identifier:** 1.3.18.0.2.12.56
- **Values:** The following ASN.1 syntax describes the BER encoding of the request value.

```
RequestValue ::= SEQUENCE {
 errorOption      INTEGER {
    retry         (1),
    display       (2),
    delete        (3) },
 failureId        OCTET STRING,
 agreementDN      LDAPDN
}
```

Where,

errorOption - An integer value indicating the operation to be performed on the advanced replication error log. If set to **retry**, tries to reprocess one or all failed replication updates. If set to **delete**, deletes one or all failed replication updates. If set to **show**, shows the failed update specified by the failureId. The failureId cannot be set to **0** when errorOption is set to **show**.

failureId - A string indicating the target of the operation. If set to **0**, then all advanced replication errors in the error log are either retried or deleted based on the errorOption setting. Otherwise, this value specifies the failure ID in the advanced replication error log that is to be retried, displayed, or deleted based on the errorOption setting. The failure ID can be determined by searching the agreementDN for the **ibm-replicationFailedChanges** operational attribute. The failure ID must be in the range 1 - 4294967295.

agreementDN - A distinguished name (DN) containing the replication agreement that this operation affects.

- **Response object identifier:** 1.3.18.0.2.12.57
- **Response description:** This extended operation response is used to return error information when a problem is encountered with performing the **Control replication error log** extended operation.
- **Response values:** The following ASN.1 syntax describes the BER encoding of the response value.

```
ResponseValue ::= SEQUENCE {
 resultCode         INTEGER,
 countEffected      OCTET STRING,
 msg                OCTET STRING
}
```

  Where,

  `resultCode` - An integer value indicating whether the extended operation is successful. Standard LDAP return codes are returned as values within this portion of the extended operation response.

  `countEffected` - A string containing the number of error log failures retried, deleted, or displayed.

  `msg` - A string containing a reason code message. In most cases, it is an error message indicating why the extended operation failed. In some cases, it can be an informational or warning message.
- **Response detailed description:**

  The following table summarizes some different error scenarios and the **Control replication error log** response returned for such scenarios.

Error scenario	Control replication error log response
`agreementDN` does not exist or is not a replication agreement	Returns an **LDAP_NO_SUCH_OBJECT** return code
`failureId` does not exist for any agreement in this backend	Returns an **LDAP_NO_SUCH_OBJECT** return code
Backend does not support advanced replication	Returns an **LDAP_UNWILLING_TO_PERFORM** return code
`failureId` is not logged to this `agreementDN`	Returns an **LDAP_UNWILLING_TO_PERFORM** return code
Not authorized to perform operation	Returns an **LDAP_INSUFFICIENT_ACCESS** return code
Syntax of DN specified is not correct	Returns an **LDAP_INVALID_DN_SYNTAX** return code
Value for the input option is not valid	Returns an **LDAP_PROTOCOL_ERROR** return code
LDAP server is unable to decode the request	Returns an **LDAP_DECODING_ERROR** return code

Control replication queue

- **Name: Control replication queue**
- **Description:** Used to indicate which pending changes in the advanced replication queue for a replication agreement ought to be skipped (deleted) and not replicated to the consumer server.
- **Assigned object identifier:** 1.3.18.0.2.12.17
- **Values:** The following ASN.1 syntax describes the BER encoding of the request value.

```
requestValue ::= SEQUENCE {
 action INTEGER {
    skipAll         (0),
    skipSingle      (1) },
 agreementDN LDAPDN,
 changeId OCTET STRING
}
```

Where,

`action` - An integer value indicating the operation that is to be performed on the advanced replication queue. If set to **skipAll**, the server skips (deletes) all updates that have not yet been replicated from the replication agreement. If set to **skipSingle**, the server skips (deletes) the specified `changeId`. Only the next change to be replicated can be skipped in this manner. If the `changeId` that is specified is not the first one in the list of pending changes, the extended operation fails. This ensures that the operation only affects the entry that is preventing advanced replication from occurring.

`agreementDN` - A distinguished name (DN) containing the replication agreement that this operation affects.

`changeID` - A string that identifies the change ID of a pending operation in the replication agreement that is to be skipped (deleted). The `changeId` that is specified must be in the range 1 - 4294967295. Change IDs can be determined by searching the `agreementDN` for the **ibm-replicationPendingChanges** operational attribute. The `changeId` is required when action is set to **skipSingle** and ignored when action is set to **skipAll**.

- **Response object identifier:** 1.3.18.0.2.12.17
- **Response description:** This extended operation response is used to return error information when a problem is encountered with performing the **Control replication queue** extended operation. If `action` is set to **skipAll** and there are no pending updates in the advanced replication queue, the extended operation is considered successful.
- **Response values:** The following ASN.1 syntax describes the BER encoding of the response value.

```
ResponseValue ::= SEQUENCE {
 resultCode         INTEGER,
 msg                OCTET STRING,
 changesSkipped     INTEGER
}
```

Where,

`resultCode` - An integer value indicating whether the extended operation is successful. Standard LDAP return codes are returned as values within this portion of the extended operation response.

`msg` - A string containing a reason code message. In most cases, it is an error message indicating why the extended operation failed. In some cases, it can be an informational or warning message.

`changesSkipped` - An integer value indicating the number of pending updates in the advanced replication queue that have been skipped (deleted).

- **Response detailed description:**

  The following table summarizes some different error scenarios and the **Control replication queue response** returned for such scenarios.

Error scenario	Control replication queue response
`agreementDN` does not exist or is not a replication agreement	Returns an **LDAP_NO_SUCH_OBJECT** return code

Error scenario	Control replication queue response
Backend does not support advanced replication	Returns an **LDAP_UNWILLING_TO_PERFORM** return code
The specified `changeId` is not the next change to be replicated	Returns an **LDAP_UNWILLING_TO_PERFORM** return code
Not authorized to perform the requested operation	Returns an **LDAP_INSUFFICIENT_ACCESS** return code
Syntax of DN specified is not correct	Returns an **LDAP_INVALID_DN_SYNTAX** return code
Value for the input option is not valid	Returns an **LDAP_PROTOCOL_ERROR** return code
LDAP server is unable to decode the request	Returns an **LDAP_DECODING_ERROR** return code

Effective password policy

- **Name: Effective password policy**
- **Description:** Used to query the effective password policy for a user or group entry and lists the policies used in determining its effective password policy.
- **Assigned object identifier:** 1.3.18.0.2.12.75
- **Values:** The following ASN.1 syntax describes the BER encoding of the request value.

```
RequestValue ::= SEQUENCE  {
 entryDN LDAPDN
}
```

  where,

  `entryDN` - A distinguished name (DN) containing the entry whose effective password policies and password policy attribute values are being queried.
- **Detailed description:** The **Effective password policy** extended operation is only allowed when bound as an LDAP root or directory data administrator, or as a user querying its own effective password policy. An LDAP root or directory data administrator is allowed to query the effective password policy of other users and groups in the directory. When a user entry is queried, this extended operation shows the effective password policy entries and values that are used to control the user's authentication and password modifications. When a group entry is queried, this extended operation provides the effective password policy that is a combination of the group's password policy attributes and the global password policy entry, **cn=pwdpolicy,cn=ibmpolicies**.
- **Response object identifier:** 1.3.18.0.2.12.77
- **Response description:** When a user entry is queried, this extended response shows the effective password entries and values used to control the user's authentication and password modifications. When a group entry is queried, this extended operation provides the effective password policy that is a combination of the group's password policy attributes and the global password policy entry, **cn=pwdpolicy,cn=ibmpolicies**. If a user is querying their own effective password policy, the `objectNames` are not returned.
- **Response values:** The following describes the response value.

```
ResponseValue ::= SEQUENCE {
  attributes    SEQUENCE OF SEQUENCE {
                 attributeType       AttributeDescription,
                 values              SET OF AttributeValue
```

```
  }
  objectNames    [0] SEQUENCE {
                 objectName         LDAPDN OPTIONAL
  }
}
```

Where,

`attributes` - The password policy attribute types and values that are contained in the user's or group's effective password policy.

`objectName` - The distinguished names of all password policy entries from where the effective password policy attribute values are derived. The `objectName` field is only returned in the extended operation response when bound as an LDAP root or directory data administrator. It is not returned when bound as a normal user.

- **Response detailed description:**

  The following table summarizes some different error scenarios and the **Effective password policy** response returned for such scenarios.

Error scenario	Effective password policy response
An unauthorized user tries to perform the extended operation	Returns an **LDAP_INSUFFICIENT ACCESS** return code
Syntax of DN specified is not correct	Returns an **LDAP_INVALID_DN_SYNTAX** return code
Insufficient memory to perform the operation	Returns an **LDAP_NO_MEMORY** return code
`entryDN` does not exist	Returns an **LDAP_NO_SUCH_OBJECT** return code
Internal server error	Returns an **LDAP_OPERATIONS_ERROR** return code
LDAP server is unable to decode the request	Returns an **LDAP_PROTOCOL_ERROR** return code
Returned for errors not covered by previously documented return codes. Check the corresponding error message for further details.	Returns an **LDAP_OTHER** return code

GetDnForUserid

- **Name: GetDnForUserid**
- **Description:** The **GetDnForUserid** extended operation is deprecated. This extended operation 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. The extended operation is rejected if the EXOP backend is not configured.
- **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 performs 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 server specified in the **IBMLdapProxyControl** to retrieve all of the specified user ID's distinguished names. The user might 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 returned for such scenarios.

Error scenario	EXOP backends 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

GetEffectiveACL

- **Name: GetEffectiveACL**
- **Description:** Retrieves the effective ACLs for a directory entry that is based on the bind identity and directory/entry access information.
- **Assigned object identifier:** 1.3.18.0.2.12.82
- **Values:** The following ASN.1 syntax describes the BER encoding of the request value.

```
RequestValue ::= SEQUENCE {
   entryDN                                LDAPDN,
   bindAuthentication                     BindAuthentication,
   bindIP                    [0]          OCTET STRING      OPTIONAL,
   timeOfDay                 [1]          OCTET STRING      OPTIONAL,
   dayOfWeek                 [2]          OCTET STRING      OPTIONAL,
```

```
   bindEncryption              [3]           BOOLEAN           OPTIONAL
}

BindAuthentication ::= CHOICE {
   simpleCramDigestBind        [0]           SimpleCramDigestBind,
   gssApiBind                  [1]           GssApiBind,
   externalBind                [2]           ExternalBind,
   anonymousBind               [3]           NULL
}

SimpleCramDigestBind ::= SEQUENCE {
   bindDN                                    LDAPDN,
   mechanism ENUMERATED {
     simple      (0),
     cramMd5     (1),
     digestMd5   (2)
   }
}

GssApiBind ::= principalRealm                  OCTET STRING

ExternalBind ::= SEQUENCE {
   subjectDN                                 LDAPDN,
   racfUserID                  [0]           OCTET STRING OPTIONAL
}
```

Where,

`entryDN` - A distinguished name (DN) of the entry that access is requested.

`bindAuthentication` - The bind identity and bind mechanism to calculate effective ACLs for.

`bindIP` - The IPv4 or IPv6 address of the client that is used to connect to the LDAP server. If `bindIP` is unspecified, it defaults to the unspecified IPv4 address, *0.0.0.0*

`timeOfDay` - The time of day the directory entry is accessed. The format is *hh:mm*, where *hh* ranges from 00 to 23, and *mm* ranges from 00 to 59. If `bindtimeOfDay` is unspecified, it defaults to the server's current time of day.

`dayOfWeek` - The day of week the directory entry is accessed. Valid values range from 0 to 6, where: *Sunday* = 0, *Monday* = 1, *Tuesday* = 2, *Wednesday* = 3, *Thursday* = 4, *Friday* = 5, *Saturday* = 6. If `binddayOfWeek` is unspecified, it defaults to the current day of week of the server.

`bindEncryption` - Indicates whether the bind DN's client connection is using encryption. If `bindEncryption` is unspecified, it defaults to FALSE.

`simpleCramDigestBind` - The bind mechanism that is used to bind to the LDAP server is a simple, CRAM-MD5, or DIGEST-MD5 bind.

`gssApiBind` - The bind mechanism that is used to bind to the LDAP server is a SASL GSS API Kerberos bind.

`externalBind` - The bind mechanism that is used to bind to the LDAP server is a SASL external bind.

`anonymousBind` - The bind mechanism that is used to bind to the LDAP server is an anonymous bind.

`bindDN` - A distinguished name (DN) used to bind to the LDAP server for a simple, CRAM-MD5, or DIGEST-MD5 bind.

`mechanism` - The mechanism that is used to bind to the LDAP server for a simple, CRAM-MD5, or DIGEST-MD5 bind.

`principalRealm` - The SASL GSS API Kerberos bind source principal that is obtained from the GSS API client credentials, which are specified as *principal@REALM*.

`subjectDN` - The subject name from the client certificate.

racfUserID - The RACF user ID associated with subjectDN. If unspecified, access is only determined for subjectDN.

- **Detailed description:** With the entryDN, optional bind, and directory entry access information, the following are returned:
  - The entry DN that access was requested.
  - The subject (bind, alternate, and group DNs) that access was calculated for.
  - The applicable attribute values (aclEntry and entryOwner) used to form the effective ACLs.
  - The effective object ACL.
  - The effective access class ACLs.
  - The effective attribute ACLs.

  Note: Administrator permissions can be restricted by ACL filters that match the administrator DN. Additionally, the ACL permissions of users who issue this extended operation can limit the results returned.
- **Response object identifier:** 1.3.18.0.2.12.82
- **Response description:** Returned by the LDAP server when it receives a **GetEffectiveACL** extended operation request.
- **Response values:** The following ASN.1 syntax describes the BER encoding of the response value.

```
ResponseValue ::= SEQUENCE {
  resultCode                          INTEGER (0..MAX),
  resultMessage                       OCTET STRING,
  entryDN                             LDAPDN,
  bindDN                              LDAPDN,
  alternateDNs                        SEQUENCE of LDAPDN,
  groupDNs                            SEQUENCE of LDAPDN,
  aclEntriesApplied                   SEQUENCE OF OCTET STRING,
  entryOwnersApplied                  SEQUENCE OF OCTET STRING,
  effectiveObjectACL                  OCTET STRING,
  effectiveAccessClassACLs            SEQUENCE OF OCTET STRING,
  effectiveAttributeACLs              SEQUENCE OF OCTET STRING
}
```

Where,

resultCode - An integer value indicating whether the extended operation is successful. Standard LDAP return codes are returned as values within this portion of the extended operation response.

resultMessage - A string containing an error message. In most cases, it indicates why the extended operation failed. In some cases, it can be an informational or warning message.

entryDN - A distinguished name (DN) of the entry that access is requested.

bindDN - The bind distinguished name that effective ACLs were calculated for.

alternateDNs - The alternate bind distinguished names that effective ACLs were calculated for.

groupDNs - The group distinguished names that effective ACLs were calculated for.

aclEntriesApplied - aclEntry attribute values that were applied to form the effective ACLs.

entryOwnersApplied - entryOwner attribute values that were applied to form the effective ACLs.

effectiveObjectACL - The object permissions (add, delete) that represent the bind identity's access to the entry DN.

effectiveAccessClassACLs - The access class permissions (read, write, search, compare) that represent the bind identity's access to the entry DN.

effectiveAttributeACLs - The attribute permissions (read, write, search, compare) that represent the bind identity's access to the entry DN.

- **Response detailed description:**

  When the LDAP server receives a **GetEffectiveACL** extended operation request, it calculates the effective access that the bind identity would have to the entry DN. Calculation of the effective access takes into account the optional bind identity and directory/entry access information. The following table summarizes some different error scenarios and the **GetEffectiveACL** returned for such scenarios.

Error scenario	GetEffectiveACL response
Successfully calculated the ACL	Returns an **LDAP_SUCCESS** return code
Cannot find the entry DN	Returns an **LDAP_NO_SUCH_OBJECT** return code
Not authorized to perform operation	Returns an **LDAP_INSUFFICIENT_ACCESS** return code
Syntax of DN specified is not correct	Returns an **LDAP_INVALID_DN_SYNTAX** return code
Value for the input option is not valid	Returns an **LDAP_PROTOCOL_ERROR** return code
LDAP server is unable to decode the request	Returns an **LDAP_DECODING_ERROR** return code
Encounters any other unsuccessful return codes	Returns the return code encountered and a detailed message describing the failure

GetPrivileges

- **Name: GetPrivileges**
- **Description:** The **GetPriveleges** extended operation is deprecated. This extended operation 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. The extended operation is rejected if the EXOP backend is not configured.
- **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 server specified in the **IBMLdapProxyControl** to retrieve all of the subject's Policy Director data.

  The following table summarizes some different error scenarios and the EXOP backend's response returned for such scenarios.

Error scenario	EXOP backends 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

Quiesce or unquiesce context

- **Name: Quiesce or unquiesce context**
- **Description:** Used to change an advanced replication context to or from a quiesced state. In a quiesced state, the entire subtree starting from the context does not accept client updates except from an LDAP administrator with the appropriate authority, if using the **Server Administration** control (OID 1.3.18.0.2.10.15), and any replication master server DNs with authority under this context. Advanced replication continues for a quiesced context. If the server is restarted, all replication contexts are then unquiesced.
- **Assigned object identifier:** 1.3.18.0.2.12.19
- **Values:** The following ASN.1 syntax describes the BER encoding of the request value.

```
RequestValue ::= SEQUENCE {
 quiesce          BOOLEAN,
 contextDN        LDAPDN
}
```

  Where,

  `quiesce` - A boolean indicating whether to quiesce (TRUE) or unquiesce (FALSE) an advanced replication context.

contextDN - A distinguished name (DN) containing the replication context that this operation affects.

- **Response object identifier:** 1.3.18.0.2.12.19
- **Response description:** This extended operation response is used to return error information when a problem is encountered with performing the **Quiesce or unquiesce context** extended operation.
- **Response values:** The following ASN.1 syntax describes the BER encoding of the response value.

```
ResponseValue ::= SEQUENCE {
 resultCode       INTEGER,
 msg              OCTET STRING
}
```

  Where,

  resultCode - An integer value indicating whether the extended operation is successful. Standard LDAP return codes are returned as values within this portion of the extended operation response.

  msg - A string containing a reason code message. In most cases, it is an error message indicating why the extended operation failed. In some cases, it can be an informational or warning message.

- **Response detailed description:**

  The following table summarizes some different error scenarios and the **Quiesce or unquiesce context** response returned for such scenarios.

Error scenario	Quiesce or unquiesce context response
contextDN does not exist or is not a replication context	Returns an **LDAP_NO_SUCH_OBJECT** return code
Backend does not support advanced replication	Returns an **LDAP_UNWILLING_TO_PERFORM** return code
Not authorized to perform operation	Returns an **LDAP_INSUFFICIENT_ACCESS** return code
Syntax of DN specified is not correct	Returns an **LDAP_INVALID_DN_SYNTAX** return code
Unable to find the request data	Returns an **LDAP_PROTOCOL_ERROR** return code
LDAP server is unable to decode the request	Returns an **LDAP_DECODING_ERROR** return code

Remote auditing

- **Name: Remote auditing**
- **Description:** Used by resource managers that do not exist on z/OS to remotely log security events in SMF.
- **Assigned object identifier:** 1.3.18.0.2.12.68
- **Values:** See Remote auditing extended operation in *z/OS IBM Tivoli Directory Server Plug-in Reference for z/OS* for more information about the **Remote auditing** extended operation request.
- **Response object identifier:** 1.3.18.0.2.12.69
- **Response description:** This extended operation response is used to return information regarding the audit log request when performing the **Remote auditing** extended operation.

- **Response values:** See Remote auditing extended operation in *z/OS IBM Tivoli Directory Server Plug-in Reference for z/OS* or more information about the **Remote auditing** extended operation response.

Remote authorization

- **Name: Remote authorization**
- **Description:** Used by resource managers that do not exist on z/OS to remotely check authorization in the z/OS Security Manager.
- **Assigned object identifier:** 1.3.18.0.2.12.66
- **Values:** See Remote authorization extended operation in *z/OS IBM Tivoli Directory Server Plug-in Reference for z/OS* for more information about the **Remote authorization** extended operation request.
- **Response object identifier:** 1.3.18.0.2.12.67
- **Response description:** This extended operation response is used to return information regarding the authorization check request when performing the **Remote authorization** extended operation.
- **Response values:** See Remote authorization extended operation in *z/OS IBM Tivoli Directory Server Plug-in Reference for z/OS* for more information about the **Remote authorization** extended operation response.

RemoteCryptoCCA

- **Name: RemoteCryptoCCA**
- **Description:** Used by remote applications to access a subset of CCA callable services provided by ICSF, allowing remote management of centralized key material stored in persistent key data sets in z/OS.
- **Assigned object identifier:** 1.3.18.0.2.12.85
- **Values:** See ICSF callable services supported by the RemoteCryptoCCA extended operation in *z/OS IBM Tivoli Directory Server Plug-in Reference for z/OS* for more information about the **RemoteCryptoCCA** extended operation request.
- **Response object identifier:** 1.3.18.0.2.12.86
- **Response description:** This extended operation response is used to return information regarding the remote crypto plug-in request when performing the **RemoteCryptoCCA** extended operation.
- **Response values:** See ICSF callable services supported by the RemoteCryptoCCA extended operation in *z/OS IBM Tivoli Directory Server Plug-in Reference for z/OS* for more information about the **RemoteCryptoCCA** extended operation response.

RemoteCryptoPKCS#11

- **Name: RemoteCryptoPKCS#11**
- **Description:** Used by applications that do not have access to a local PKCS #11 implementation and want to manage their PKCS #11 tokens and objects in z/OS.
- **Assigned object identifier:** 1.3.18.0.2.12.83
- **Values:** See ICSF callable services supported by the RemoteCryptoPKCS#11 extended operation in *z/OS IBM Tivoli Directory Server Plug-in Reference for z/OS* for more information about the **RemoteCryptoPKCS#11** extended operation request.
- **Response object identifier:** 1.3.18.0.2.12.84

- **Response description:** This extended operation response is used to return information regarding the remote crypto plug-in request when performing the **RemoteCryptoPKCS#11** extended operation.
- **Response values:** See ICSF callable services supported by the RemoteCryptoPKCS#11 extended operation in *z/OS IBM Tivoli Directory Server Plug-in Reference for z/OS* for more information about the **RemoteCryptoPKCS#11** extended operation response.

Replication topology

- **Name: Replication topology**
- **Description:** Used to synchronize replication topology-related entries across the replication topology. A replication topology entry is an entry that contains an objectclass whose name begins with **ibm-replica**, such as **ibm-replicationContext, ibm-replicaGroup, ibm-replicaSubentry, ibm-replicationAgreement**, and **ibm-replicationCredentialsSimple**. This extended operation is cascaded through all forwarding and gateway servers if `agreementDN` is not specified. If `agreementDN` is specified, then the extended operation only synchronizes the replication topology entries in the consumer server defined by the replication agreement.

  Note: Replication topology entries containing credentials are also replicated if they are located under the replication context. These entries should be located instead under the **cn=localhost** suffix.
- **Assigned object identifier:** 1.3.18.0.2.12.54
- **Values:** The following ASN.1 syntax describes the BER encoding of the request value.

```
RequestValue ::= SEQUENCE {
 contextDN                  LDAPDN,
 timeout                    INTEGER,
 agreementDN                LDAPDN OPTIONAL
}
```

  Where,

  `contextDN` - A distinguished name (DN) containing the replication context on the supplier server that the advanced replication topology-related entries are synchronized.

  `timeout` - An integer value indicating the number of seconds that the extended operation must successfully complete. If not present, or 0, the operation has an indefinite amount of time to complete.

  `agreementDN` - A distinguished name (DN) containing the replication agreement used to connect to a consumer server. If a value is specified, only the consumer server defined in the `agreementDN` is synchronized. The extended operation is not cascaded to any consumers of that server.
- **Response object identifier:** 1.3.18.0.2.12.55
- **Response description:** This extended operation response is used to return error information when a problem is encountered with performing the **Replication topology** extended operation.
- **Response values:** The following ASN.1 syntax describes the BER encoding of the response value.

```
ResponseValue ::= SEQUENCE {
 resultCode        INTEGER,
 msg               OCTET STRING,
 supplier          OCTET STRING,
 consumer          OCTET STRING
}
```

Where,

`resultCode` - An integer value indicating whether the extended operation is successful. Standard LDAP return codes are returned as values within this portion of the extended operation response.

`msg` - A string containing a reason code message. In most cases, it is an error message indicating why the extended operation failed. In some cases, it can be an informational or warning message.

`supplier` - A string containing the shortened host name and advanced replication server ID of the supplier server targeted by the extended operation. The format is '*shortenedHostName*:*serverID*'. If the shortened host name cannot be determined, the format is 'server ID:*serverID*'.

`consumer` - A string containing the *host*:*port* of the consumer server that is reporting an error. This is returned when the consumer has certain problems performing the requested operation.

- **Response detailed description:**

  The following table summarizes some different error scenarios and the **Replication topology response** returned for such scenarios.

Error scenario	Replication topology response
`contextDN` does not exist or is not a replication context	Returns an **LDAP_NO_SUCH_OBJECT** return code
`agreementDN` does not exist or is not a replication agreement	Returns an **LDAP_NO_SUCH_OBJECT** return code
Backend does not support advanced replication	Returns an **LDAP_UNWILLING_TO_PERFORM** return code
`agreementDN` is not under a replication context	Returns an **LDAP_UNWILLING_TO_PERFORM** return code
Not authorized to perform operation	Returns an **LDAP_INSUFFICIENT_ACCESS** return code
Operation did not complete with specified time	Returns an **LDAP_TIMEOUT** return code
Syntax of DN specified is not correct	Returns an **LDAP_INVALID_DN_SYNTAX** return code
Value for the input option is not valid	Returns an **LDAP_PROTOCOL_ERROR** return code
LDAP server is unable to decode the request	Returns an **LDAP_DECODING_ERROR** return code

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 directory 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 directory operations on the connection.

  The LDAP server responds 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 might 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 returns 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 servers response returned for such scenarios.

Error scenario	Server response
Server accepts connection and can handle the request	Returns an **LDAP_SUCCESS** return code
SSL/TLS is not configured	Returns an **LDAP_UNAVAILABLE** 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

unloadRequest

- **Name: unloadRequest**
- **Description:** Causes the LDAP server to unload entries from a directory into a file in LDIF format. This extended operation is used by the **ds2ldif** utility. See "ds2ldif utility" on page 225 for more information about **ds2ldif**.
- **Assigned object identifier:** 1.3.18.0.2.12.62
- **Values:** The following ASN.1 syntax describes the BER encoding of the request value:

```
RequestValue ::= SEQUENCE {
    tagOption                       BOOLEAN,
    outputFileName                  OCTET STRING,
    subtreeDN                 [0]   LDAPDN OPTIONAL,
```

```
    backendName                 [1]   OCTET STRING OPTIONAL,
    genealogicalOrder           [2]   BOOLEAN OPTIONAL,
    filterDN                    [3]   LDAPDN OPTIONAL,
    noControlValues             [4]   BOOLEAN OPTIONAL,
    unloadLocalhost             [5]   BOOLEAN OPTIONAL
}
```

Where,

`tagOption` - A boolean indicating whether the encryption tag of **userPassword** attribute values should be displayed in the clear when directory data is unloaded. This value corresponds to the **–t** option of the **ds2ldif** utility.

`outputFileName` - A string containing the fully qualified z/OS UNIX System Services file name or data set name that the server writes the unloaded directory entries in LDIF format. This value corresponds to the **–o** option of the **ds2ldif** utility.

`subtreeDN` - A distinguished name (DN) containing the name of the top entry to unload. This entry and all of the entries below it in the directory hierarchy are unloaded. The value must be the DN of an entry in an LDBM, TDBM, or CDBM backend or of the LDAP server schema entry, cn=schema. This value corresponds to the **-s** option of the **ds2ldif** utility.

`backendName` - A string containing the name of an LDBM, TDBM, or CDBM backend. All the entries in the backend are unloaded. The value is either the optional fourth parameter of the **database** option in the LDAP server configuration file or the name generated for the backend by the LDAP server if the parameter is not specified in the configuration file. This value corresponds to the **-n** option of the **ds2ldif** utility.

`genealogicalOrder` - A boolean value that specifies how the entries are unloaded. If set to TRUE, entries in each subtree are unloaded together, doing a depth-first traversal of the directory. If set to FALSE or if not specified, there is no guaranteed order of unloaded entries, other than parent entries are always unloaded before child entries. This value corresponds to the **-g** option of the **ds2ldif** utility.

`filterDN` - A distinguished name (DN) of a replication filter entry that contains **ibm-replicationFilterAttr** attribute values. These values are filters used to skip entire entries or attributes within entries while unloading the directory. See “Partial replication” on page 536 for more information about replication filter entries. Advanced replication must be enabled in the LDAP server with the CDBM backend configured and **useAdvancedReplication on** specified in the CDBM backend section of the server configuration file. This value corresponds to the **-q** option of the **ds2ldif** utility.

`noControlValues` - A boolean indicating whether the **replicateOperationalAttributes** control value is unloaded for each entry. If set to TRUE, the **replicateOperationlAttributes** control is not unloaded for each entry. If set to FALSE or if not specified and the server compatibility level is 5 or greater, the **replicateOperationalAttributes** control is unloaded for each entry. The **replicateOperationalAttributes** control value contains the **modifyTimestamp, createTimestamp, creatorsName**, and **modifiersName** attribute types and values for the entry base64 encoded. See “serverCompatLevel {3 | 4 | 5 | 6 | 7| 8}” on page 125 for more information about server compatibility. This value corresponds to the **-j** option of the **ds2ldif** utility.

`unloadLocalHost` - A boolean indicating whether the **cn=localhost** subtree is unloaded if it exists on the targeted LDAP server. This boolean can only be set to TRUE when `subtreeDN` is not specified. This value corresponds to the **-l** option of the **ds2ldif** utility.

- **Detailed description:** The **unloadRequest** extended operation is rejected if the requester is not bound as an LDAP root, directory data, or schema (only when unloading the schema entry) administrator. `subtreeDN` and `backendName` cannot both be specified in the request. If neither is specified and there is only one LDBM, TDBM, or CDBM backend configured, then that backend is unloaded; otherwise, an error is returned.

  The **aclEntry, aclPropagate, entryOwner**, and **ownerPropagate** attributes are included in an unloaded entry if these attributes have been explicitly set for that entry. The unloaded entry also includes the **ibm-entryuuid** attribute. The **replicateOperationalAttributes** control value is written to the output LDIF file if `noControlValues` is set to FALSE.

  The **ds2ldif** utility uses this extended operation to unload the selected directory data if it encounters a problem starting the backend that is to be unloaded or if the **-r** command-line option is specified. For additional details about the operation of the **ds2ldif** utility, see "ds2ldif utility" on page 225.
- **Response object identifier:** 1.3.18.0.2.12.63
- **Response description:** The response indicates the number of entries that were successfully unloaded by the LDAP server as a result of the **unloadRequest** extended operation.
- **Response values:** The following ASN.1 syntax describes the BER encoding of the response value.

```
ResponseValue :== SEQUENCE {
   entryCount INTEGER
}
```

- **Response detailed description:**

  The following table summarizes some different error scenarios and the **unloadRequest response** returned for such scenarios:

Error scenario	unloadRequest response
Not bound as an LDAP administrator	Returns an **LDAP_OTHER** return code
Multiple LDBM, TDBM, or CDBM backends are active in the server but a `subtreeDN` or `backendName` is not specified	Returns an **LDAP_OTHER** return code
A non-existent `subtreeDN`, `filterDN`, or `backendName` is specified	Returns an **LDAP_OTHER** return code
Error opening or writing the output LDIF file	Returns an **LDAP_OTHER** return code
Both `subtreeDN` and `backendName` are specified	Returns an **LDAP_PROTOCOL_ERROR** return code
A zero-length value is specified for `subtreeDN`, `filterDN`, or `backendName`	Returns an **LDAP_PROTOCOL_ERROR** return code
LDAP server is unable to parse the request	Returns an **LDAP_PROTOCOL_ERROR** return code
A `subtreeDN` is specified when `unloadLocalHost` is set to TRUE	Returns an **LDAP_PROTOCOL_ERROR** return code
`filterDN` specified when CDBM backend is not configured or **useAdvancedReplication off** is specified in the CDBM backend section of the server configuration file	Returns an **LDAP_UNWILLING_TO_PERFORM** return code

User type

- **Name: User type**
- **Description:** Used to query the user type and roles defined for the user performing the request.
- **Assigned object identifier:** 1.3.18.0.2.12.37
- **Values:** None
- **Detailed description:** The **User type** extended operation allows the requesting user to determine their user type and one or more roles. User types returned can be one of the following:
  - **Root administrator** - The user is the LDAP root administrator defined in the configuration file (**adminDN** configuration option) or is defined in the administrative group entry and assigned the root administrator role.
  - **Administrative group member** - The user is defined in the administrative group member entry under **cn=admingroup,cn=configuration**.
  - **Local OS user** - The user bound to the LDAP server using a user name and authenticated to the z/OS Security Manager.
  - **LDAP user Type** - The user's credentials are stored in the CDBM, TDBM, and LDBM backend of the LDAP server.
  - **Master server DN** - The user is defined as a Master Server DN for replication.
  - **SAF administrative group member** - The user is a member of the **cn=safadmingroup,cn=configuration** entry.

  User roles returned can be one or more of the following:
  - **Directory data administrator**
  - **No administrator**
  - **Operational administrator**
  - **Password administrator**
  - **Replication administrator**
  - **Schema administrator**
  - **Server configuration group member**
  - **LDAP user role** - The user's access to directory data is controlled only by ACLs.

  See Chapter 9, "Administrative group and roles," on page 159 for more information about user roles.
- **Response object identifier:** 1.3.18.0.2.12.38
- **Response description:** This response is used to return the user type and one or more roles for the user issuing the request.
- **Response values:** The following describes the response value.

```
ResponseValue ::= SEQUENCE {
 userType          OCTET STRING,
 numberOfRoles     INTEGER,
 roles             SEQUENCE OPTIONAL {
    userRole          OCTET STRING
 }
}
```

where,

`userType` - A string containing one of the user's type. It has one of the following values: **root_administrator, admin_group_member, local_os_user, ldap_user_type, master_server_dn**, and **saf_admin_group_member**.

`numberOfRoles` - An integer containing the number of user roles provided in the next sequence.

`userRole` - A string value contained in a sequence of string values which indicates the user's role or roles. Each entry in the sequence has one of the following values: **directory_data_administrator, no_administrator, operational_administrator, password_administrator, replication_administrator, schema_administrator, server_config_group_member**, and **ldap_user_role**.

- **Response detailed description:**

  The following table summarizes some different error scenarios and the **User type** response returned for such scenarios.

Error scenario	User type response
LDAP server unable to decode the request	Returns an **LDAP_PROTOCOL_ERROR** return code
Returned for errors not covered by previously documented return code. Check the corresponding error message for further details.	Returns an **LDAP_OTHER** return code

Appendix E. SMF records

SMF Record Type 83, subtype 3 records

When auditing is enabled for LDAP events, SMF record type 83, subtype 3 records are recorded in the SMF data set. Each logged LDAP event has a unique event code with a corresponding event code qualifier. The event qualifier indicates whether the event succeeded or failed. For more information about SMF records and the relocates that are common to all SMF Type 83 subtype 2 and above records, see *z/OS Security Server RACF Macros and Interfaces*.

Table 89 shows LDAP event codes and event code qualifiers:

Table 89. LDAP event codes

Event	Operation	Event qualifier	Description	Relocate type sections
1	Add	0	Successful LDAP operation	Common relocates (100-114) 201-208
		1	Failed LDAP operation	
2	Bind	0	Successful LDAP operation	Common relocates (100-114) 201-207 209
		1	Failed LDAP operation	
3	Compare	0	Successful LDAP operation	Common relocates (100-114) 201-207 210
		1	Failed LDAP operation	
4	Connect	0	Successful LDAP operation	Common relocates (100-114)
5	Delete	0	Successful LDAP operation	Common relocates (100-114) 201-207
		1	Failed LDAP operation	
6	Disconnect	0	Successful LDAP operation	Common relocates (100-114), 223
7	Extended Operation	0	Successful LDAP operation	Common relocates (100-114) 201-207 221
		1	Failed LDAP operation	
8	Modify	0	Successful LDAP operation	Common relocates (100-114) 201-207 211-213
		1	Failed LDAP operation	
9	Modify DN	0	Successful LDAP operation	Common relocates (100-114) 201-207 214-216
		1	Failed LDAP operation	
10	Search	0	Successful LDAP operation	Common relocates (100-114) 201-207 218-220
		1	Failed LDAP operation	
11	Unbind	0	Successful LDAP operation	Common relocates (100-114)
12	Abandon	0	Successful LDAP operation	Common relocates (100-114) 201-207, 222

Table 90 shows LDAP extended relocates:

Table 90. LDAP extended relocates

Relocate #	Field name	Type	Length	Audited by event code	Description
100	LDAP_RESERVED_01	CHAR	16	N/A	Reserved
101	LDAP_REQ_TIMESTP	CHAR	16	1, 2, 3, 5, 7, 8, 9, 10, 11	The elapsed time for request completion (in microseconds).

Table 90. LDAP extended relocates (continued)

Relocate #	Field name	Type	Length	Audited by event code	Description
102	LDAP_SERVER_URL	CHAR	32	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	The server's IPv4 or IPv6 address and port number (in LDAP URL format) that processed the client connection.
103	LDAP_CONN_ID	CHAR	8	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	The internal connection ID used to indicate operations performed on the same client connection.
104	LDAP_MESSAGE_ID	CHAR	8	1, 2, 3, 5, 7, 8, 9, 10, 11	The message ID from the BER encoded data from the client that is used to connect events.
105	LDAP_BIND_DN	CHAR	512	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	The bind DN for the connection.
106	LDAP_CLIENT_SECL	CHAR	32	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	The LDAP client security label, if there is one.
107	LDAP_SRC_IP_ADDR	CHAR	16	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	The IP address of the client performing the request. If set to **PC**, the request came across the PC interface. It is set to **XCF** for sysplex replica requests and to **SLAPI** for internal SLAPI requests.
108	LDAP_AUDIT_VERSION	CHAR	2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	The version of the LDAP audit support in use. It is always set to **V1**.
109	LDAP_EVENT_CODE	CHAR	2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	The LDAP event number that is associated with this SMF audit type 83 subtype-3 record. The LDAP event numbers range from 1 to 11.
110	LDAP_RESERVED_04	CHAR	227	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	Reserved
111	LDAP_MAPCERT_OPT	CHAR	2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	Represents the setting of the **sslMapCertificate** configuration option. See note 1 on page 723 for more information.
112	LDAP_MAPPED_SAFID	CHAR	8	1, 2, 3, 5, 7, 8, 9, 10, 11	The SAF ID, if any, associated with the bind DN.
113	LDAP_POLICY_UPDATED	CHAR	1	2	**T** (true) or **F** (false) if the password policy operational attribute values in the bind DN entry are updated.
114	LDAP_FLAGS	CHAR	10	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	Character representation of the hex value for flag settings. See note 2 on page 724 for more information.
201	LDAP_PROTOCOL_VER	CHAR	2	1, 2, 3, 5, 7, 8, 9, 10, 11	The LDAP protocol version set to **2** or **3**.
202	LDAP_RETURN_CODE	INT	10	1, 2, 3, 5, 7, 8, 9, 10	The LDAP server return code in decimal (blank if there was no return code provided).
203	LDAP_ERROR_MSG	CHAR	256	1, 2, 3, 5, 7, 8, 9, 10	The reason code message number and text, if there is one, sent from the server to the client.
204	LDAP_ENTRY_NM	CHAR	512	1, 2, 3, 5, 7, 8, 9, 10	The target entry of the requested operation (bind DN, DN to be deleted, DN to be added, and so on).
205	LDAP_RELOC_REQ_TS	CHAR	32	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	The time and date the request was received in the following format: `Tue Oct 13 16:36:25 2009`
206	LDAP_ENTRY_SUFFIX	CHAR	64	1, 2, 3, 5, 7, 8, 9, 10	The normalized DN of the backend suffix that handled the request.
207	LDAP_CNTRLS_PRESNT	CHAR	512	1, 2, 3, 5, 7, 8, 9, 10, 11	The control OID and the criticality of the control in the request. Each control in the request has its own relocate. It has the following format: `ControlOID,TRUE|FALSE`
208	LDAP_ADD_ATTR	CHAR	64 (maximum 1299)	1	The name of the attribute added in the add request. Each attribute type added in the add request has its own relocate and is separated by 1 space. There is a maximum length of 1299 bytes available for all attribute types added in the add request.

Table 90. LDAP extended relocates (continued)

Relocate #	Field name	Type	Length	Audited by event code	Description
209	LDAP_BIND_MECH	CHAR	16	2	Indicates the authentication method using during bind. It has one of the following values: **SIMPLE, EXTERNAL, GSSAPI, DIGEST-MD5**, or **CRAM-MD5**
210	LDAP_COMPARE_ATTR	CHAR	64	3	The name of the attribute being compared.
211	LDAP_MOD_ATTR_DEL	CHAR	64 (maximum 909)	8	The name of the attribute that is deleted in the modify request. Each attribute type that is deleted in the modify request has its own relocate and is separated by 1 space. There is a maximum length of 909 bytes available for all attribute types that are deleted in the modify request.
212	LDAP_MOD_ATTR_ADD	CHAR	64 (maximum 909)	8	The name of the attribute added in the modify request. Each attribute type added in the modify request has its own relocate and is separated by 1 space. There is a maximum length of 909 bytes available for all attribute types added in the modify request.
213	LDAP_MOD_ATTR_REP	CHAR	64 (maximum 909)	8	The name of the attribute that is replaced in the modify request. Each attribute type that is replaced in the modify request has its own relocate and is separated by 1 space. There is a maximum length of 909 bytes available for all attribute types that are replaced in the modify request.
214	LDAP_MDN_NEW_RDN	CHAR	32	9	The new relative distinguished name for the target entry in the **modifydn** request.
215	LDAP_MDN_DOLD_RDN	CHAR	1	9	**T** (true) or **F** (false) if the old relative distinguished name (RDN) is deleted in the **modifydn** request.
216	LDAP_MDN_NEW_SUP	CHAR	512	9	The new superior, if there is one, of the target entry that is specified in the **modifydn** request.
218	LDAP_SEARCH_FILTER	CHAR	256	10	The search filter that is specified on the search request.
219	LDAP_SEARCH_SCOPE	CHAR	12	10	The scope of the search request. It is set to **SUBTREE** for subtree level searches, **ONE** for one level searches, and **BASE** for base level searches.
220	LDAP_SEARCH_ATTRS	CHAR	64 (maximum 833)	10	The name of the attribute that is requested to be returned on the search request. Each attribute type that is requested in the search request has its own relocate and is separated by 1 space. There is a maximum length of 833 bytes available for all requested attribute types in the search request.
221	LDAP_XOP_REQ_NAME	CHAR	64	7	The OID of the extended operation request.
222	LDAP_TARGET_MESSAGE_ID	CHAR	8	12	The abandon target message ID from the BER encoded data from the client that is used to match the abandon and the request that is being abandoned.
223	LDAP_DISCONNECT_CAUSE	CHAR	10	6	Indicates the reason of a disconnect event.

Note:

1. The LDAP_MAPCERT_OPT value represents the value of the **sslMapCertificate** option in the LDAP server configuration file:
   - **00** not an external bind request or **sslMapCertificate off ignore/fail** specified
   - **01** **sslMapCertificate check ignore**
   - **02** **sslMapCertificate add ignore**
   - **03** **sslMapCertificate replace ignore**
   - **11** **sslMapCertificate check fail**

12 sslMapCertificate add fail
13 sslMapCertificate replace fail

2. The LDAP_FLAGS value is the character representation of the hex value for the following possible flag settings:

```
ADMIN_ROLE_NONE                 0x00000001
ADMIN_ROLE_ROOT                 0x00000002
ADMIN_ROLE_DIR                  0x00000004
ADMIN_ROLE_REPL                 0x00000008
ADMIN_ROLE_SCHEMA               0x00000010
ADMIN_ROLE_CONFIG               0x00000020
ADMIN_ROLE_PASSWORD             0x00000040
ADMIN_ROLE_OPER                 0x00000080
ADMIN_ROLE_SAF                  0x00000100
```

   where, the bound user has one or more of the following administrative roles,
   ADMIN_ROLE_NONE=No administrator
   ADMIN_ROLE_ROOT=Root administrator
   ADMIN_ROLE_DIR=Directory data administrator
   ADMIN_ROLE_REPL=Replication administrator
   ADMIN_ROLE_SCHEMA=Schema administrator
   ADMIN_ROLE_CONFIG=Server configuration group member
   ADMIN_ROLE_PASSWORD=Password administrator
   ADMIN_ROLE_OPER=Operational administrator
   ADMIN_ROLE_SAF=The administrative user is a member of **cn=safadmingroup,cn=configuration** and the roles are defined in RACF.

   Example: If the user is a member of **cn=safadmingroup,cn=configuration** and has been permitted to all administrative roles, `0x000001FF` is in this field.

3. The LDAP_DISCONNECT_CAUSE value is the character representation of the hex value for the following possible flag settings:

```
DISCONNECT_TIMEOUT              0x00000001
DISCONNECT_NIC_FAILED           0x00000002
DISCONNECT_CLOSE_ABNORMAL       0x00000004
DISCONNECT_CLOSE_CLIENT         0x00000008
DISCONNECT_MESSAGE_INVALID      0x00000010
DISCONNECT_ERROR_INTERNAL       0x00000020
DISCONNECT_ERROR_WRITE          0x00000040
DISCONNECT_ERROR_SSL            0x00000080
```

   where,
   DISCONNECT_TIMEOUT=Idle connection timeout.
   DISCONNECT_NIC_FAILED=Network interface failed.
   DISCONNECT_CLOSE_ABNORMAL=Connection closed abnormally.
   DISCONNECT_CLOSE_CLIENT=Client closed connection.
   DISCONNECT_MESSAGE_INVALID=The request is not a valid LDAP message.
   DISCONNECT_ERROR_INTERNAL=Internal error. Such as, out of storage error, unable to encode or decode LDAP message.
   DISCONNECT_ERROR_WRITE=Failed to write LDAP message to network connection.
   DISCONNECT_ERROR_SSL=Error occurred in System SSL.

   Note: There might be two values that are combined. For example: DISCONNECT_TIMEOUT and DISCONNECT_CLOSE_CLIENT can occur simultaneously and the value is 0x00000009.

RACF SMF unload utility output

A general description of the format of the tabular records that are created by the SMF unload utility can be found in *z/OS Security Server RACF Macros and Interfaces*, in the topic "The format of the unloaded SMF type83 data".

The data following the header in the tabular record varies according to the LDAP event that was recorded. The LDAP event types are:

Table 91. Event type strings

Event code from SMF type 83 subtype 3 records	Tabular output Event type strings
1	*ADD
2	*BIND
3	*COMPARE
4	*CONN
5	*DELETE
6	*DISCONN
7	*EXOP
8	*MODIFY
9	*MODDN
10	*SEARCH
11	*UNBIND
12	*ABANDON

Event qualifiers are the same for all LDAP event codes:

Table 92. Event qualifiers

Event qualifier	Event qualifier number	Event description
SUCCESS	0	Audit of successful LDAP operation
FAILURE	1	Audit of failed LDAP operation

Table 93. Event specific fields for LDAP add event (Event code 1)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_RESERVED_01	CHAR	16	3000	3015	Reserved.
LDAP_REQ_TIMESTP	CHAR	16	3018	3033	The elapsed time for request completion (in microseconds).
LDAP_SERVER_URL	CHAR	32	3036	3067	The server's IPv4 or IPv6 address and port number (in LDAP URL format) that processed the client connection.
LDAP_CONN_ID	CHAR	8	3070	3077	The internal connection ID used to indicate operations performed on the same client connection.
LDAP_MESSAGE_ID	CHAR	8	3080	3087	The message ID from the BER encoded data from the client which is used to connect events.
LDAP_BIND_DN	CHAR	512	3090	3601	The bind DN for the connection.
LDAP_CLIENT_SECL	CHAR	32	3604	3635	The LDAP client security label, if there is one.

Table 93. Event specific fields for LDAP add event (Event code 1) (continued)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_SRC_IP_ADDR	CHAR	16	3638	3653	The IP address of the client performing the request. If set to **PC**, the request came across the PC interface. It is set to **XCF** for sysplex replica requests and to **SLAPI** for internal SLAPI requests.
LDAP_AUDIT_VERSION	CHAR	2	3656	3657	The version of the LDAP audit support in use. It is always set to **V1**.
LDAP_EVENT_CODE	CHAR	2	3660	3661	Event number 1 (add)
LDAP_MAPCERT_OPT	CHAR	2	3664	3665	Value is always 00
LDAP_MAPPED_SAFID	CHAR	8	3668	3675	The SAF ID, if any, associated with the bind DN.
LDAP_POLICY_UPDATED	CHAR	1	3678	3678	Blank for this operation.
LDAP_FLAGS	CHAR	10	3681	3690	Character representation of the hex value for flag settings. See note 2 on page 724 for more information.
LDAP_RESERVED_04	CHAR	227	3693	3919	Reserved
LDAP_PROTOCOL_VER	CHAR	2	3922	3923	The LDAP protocol version set to **2** or **3**.
LDAP_RETURN_CODE	INT	10	3926	3935	The LDAP server return code in decimal (blank if there was no return code provided).
LDAP_ERROR_MSG	CHAR	256	3938	4193	The reason code message number and text, if there is one, sent from the server to the client.
LDAP_ENTRY_NM	CHAR	512	4196	4707	The target entry of the requested operation (bind DN, DN to be deleted, DN to be added, and so on).
LDAP_RELOC_REQ_TS	CHAR	32	4710	4741	The time and date the request was received in the following format: `Tue Oct 13 16:36:25 2009`
LDAP_ENTRY_SUFFIX	CHAR	64	4744	4807	The normalized DN of the backend suffix that handled the request.
LDAP_CNTRLS_PRESNT	CHAR	512	4810	5321	The control OID and the criticality of the control in the request. Each control in the request has its own relocate. It has the following format: `ControlOID,TRUE|FALSE`
LDAP_ADD_ATTR	CHAR	64 (maximum 1299)	5324	6622	The name of the attribute added in the add request. Each attribute type added in the add request has its own relocate and is separated by 1 space. There is a maximum length of 1299 bytes available for all attribute types added in the add request.

Table 94. Event specific fields for LDAP bind event (Event code 2)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_RESERVED_01	CHAR	16	3000	3015	Reserved
LDAP_REQ_TIMESTP	CHAR	16	3018	3033	The elapsed time for request completion (in microseconds).
LDAP_SERVER_URL	CHAR	32	3036	3067	The server's IPv4 or IPv6 address and port number (in LDAP URL format) that processed the client connection.
LDAP_CONN_ID	CHAR	8	3070	3077	The internal connection ID used to indicate operations performed on the same client connection.
LDAP_MESSAGE_ID	CHAR	8	3080	3087	The message ID from the BER encoded data from the client which is used to connect events.
LDAP_BIND_DN	CHAR	512	3090	3601	The bind DN for the connection.

Table 94. Event specific fields for LDAP bind event (Event code 2) (continued)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_CLIENT_SECL	CHAR	32	3604	3635	The LDAP client security label, if there is one.
LDAP_SRC_IP_ADDR	CHAR	16	3638	3653	The IP address of the client performing the request. If set to **PC**, the request came across the PC interface. It is set to **XCF** for sysplex replica requests and to **SLAPI** for internal SLAPI requests.
LDAP_AUDIT_VERSION	CHAR	2	3656	3657	The version of the LDAP audit support in use. It is always set to **V1**.
LDAP_EVENT_CODE	CHAR	2	3660	3661	Event number 2 (bind)
LDAP_MAPCERT_OPT	CHAR	2	3664	3665	Represents the setting of the **sslMapCertificate** configuration option. See note 1 on page 723 for more information.
LDAP_MAPPED_SAFID	CHAR	8	3668	3675	The SAF ID, if any, associated with the bind DN.
LDAP_POLICY_UPDATED	CHAR	1	3678	3678	**T** (true) or **F** (false) if the password policy operational attribute values in the bind DN entry are updated.
LDAP_FLAGS	CHAR	10	3681	3690	Character representation of the hex value for flag settings. See note 2 on page 724 for more information.
LDAP_RESERVED_04	CHAR	227	3693	3919	Reserved
LDAP_PROTOCOL_VER	CHAR	2	3922	3923	The LDAP protocol version set to **2** or **3**.
LDAP_RETURN_CODE	INT	10	3926	3935	The LDAP server return code in decimal (blank if there was no return code provided).
LDAP_ERROR_MSG	CHAR	256	3938	4193	The reason code message number and text, if there is one, sent from the server to the client.
LDAP_ENTRY_NM	CHAR	512	4196	4707	The target entry of the requested operation (bind DN, DN to be deleted, DN to be added, and so on).
LDAP_RELOC_REQ_TS	CHAR	32	4710	4741	The time and date the request was received in the following format: `Tue Oct 13 16:36:25 2009`
LDAP_ENTRY_SUFFIX	CHAR	64	4744	4807	The normalized DN of the backend suffix that handled the request.
LDAP_CNTRLS_PRESNT	CHAR	512	4810	5321	The control OID and the criticality of the control in the request. Each control in the request has its own relocate. It has the following format: `ControlOID,TRUE|FALSE`
LDAP_BIND_MECH	CHAR	16	5324	5339	Indicates the authentication method using during bind. It has one of the following values: **SIMPLE, EXTERNAL, GSSAPI, DIGEST-MD5**, or **CRAM-MD5**

Table 95. Event specific fields for LDAP compare event (Event code 3)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_RESERVED_01	CHAR	16	3000	3015	Reserved
LDAP_REQ_TIMESTP	CHAR	16	3018	3033	The elapsed time for request completion (in microseconds).
LDAP_SERVER_URL	CHAR	32	3036	3067	The server's IPv4 or IPv6 address and port number (in LDAP URL format) that processed the client connection.
LDAP_CONN_ID	CHAR	8	3070	3077	The internal connection ID used to indicate operations performed on the same client connection.

Table 95. Event specific fields for LDAP compare event (Event code 3) (continued)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_MESSAGE_ID	CHAR	8	3080	3087	The message ID from the BER encoded data from the client which is used to connect events.
LDAP_BIND_DN	CHAR	512	3090	3601	The bind DN for the connection.
LDAP_CLIENT_SECL	CHAR	32	3604	3635	The LDAP client security label, if there is one.
LDAP_SRC_IP_ADDR	CHAR	16	3638	3653	The IP address of the client performing the request. If set to **PC**, the request came across the PC interface. It is set to **XCF** for sysplex replica requests and to **SLAPI** for internal SLAPI requests.
LDAP_AUDIT_VERSION	CHAR	2	3656	3657	The version of the LDAP audit support in use. It is always set to **V1**.
LDAP_EVENT_CODE	CHAR	2	3660	3661	Event number 3 (compare)
LDAP_MAPCERT_OPT	CHAR	2	3664	3665	Value is always 00
LDAP_MAPPED_SAFID	CHAR	8	3668	3675	The SAF ID, if any, associated with the bind DN.
LDAP_POLICY_UPDATED	CHAR	1	3678	3678	**T** (true) or **F** (false) if the password policy operational attribute values in the bind DN entry are updated (blank if the attribute that is being compared is not **userPassword**).
LDAP_FLAGS	CHAR	10	3681	3690	Character representation of the hex value for flag settings. See note 2 on page 724 for more information.
LDAP_RESERVED_04	CHAR	227	3693	3919	Reserved
LDAP_PROTOCOL_VER	CHAR	2	3922	3923	The LDAP protocol version set to **2** or **3**.
LDAP_RETURN_CODE	INT	10	3926	3935	The LDAP server return code in decimal (blank if there was no return code provided).
LDAP_ERROR_MSG	CHAR	256	3938	4193	The reason code message number and text, if there is one, sent from the server to the client.
LDAP_ENTRY_NM	CHAR	512	4196	4707	The target entry of the requested operation (bind DN, DN to be deleted, DN to be added, and so on).
LDAP_RELOC_REQ_TS	CHAR	32	4710	4741	The time and date the request was received in the following format: `Tue Oct 13 16:36:25 2009`
LDAP_ENTRY_SUFFIX	CHAR	64	4744	4807	The normalized DN of the backend suffix that handled the request.
LDAP_CNTRLS_PRESNT	CHAR	512	4810	5321	The control OID and the criticality of the control in the request. Each control in the request has its own relocate. It has the following format: `ControlOID,TRUE|FALSE`
LDAP_COMPARE_ATTR	CHAR	64	5324	5387	The name of the attribute being compared.

Table 96. Event specific fields for LDAP connect, delete, disconnect, and unbind events (Event code 4, 5, 6, 11)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_RESERVED_ 01	CHAR	16	3000	3015	Reserved
LDAP_REQ_TIMESTP	CHAR	16	3018	3033	The elapsed time for request completion (in microseconds). Not used for event codes 4 and 6.
LDAP_SERVER_URL	CHAR	32	3036	3067	The server's IPv4 or IPv6 address and port number (in LDAP URL format) that processed the client connection.
LDAP_CONN_ID	CHAR	8	3070	3077	The internal connection ID used to indicate operations performed on the same client connection.

Table 96. Event specific fields for LDAP connect, delete, disconnect, and unbind events (Event code 4, 5, 6, 11) (continued)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_MESSAGE_ID	CHAR	8	3080	3087	The message ID from the BER encoded data from the client which is used to connect events. Not used for event codes 4 and 6.
LDAP_BIND_DN	CHAR	512	3090	3601	The bind DN for the connection.
LDAP_CLIENT_SECL	CHAR	32	3604	3635	The LDAP client security label, if there is one.
LDAP_SRC_IP_ADDR	CHAR	16	3638	3653	The IP address of the client performing the request. If set to **PC**, the request came across the PC interface. It is set to **XCF** for sysplex replica requests and to **SLAPI** for internal SLAPI requests.
LDAP_AUDIT_VERSION	CHAR	2	3656	3657	The version of the LDAP audit support in use. It is always set to **V1**.
LDAP_EVENT_CODE	CHAR	2	3660	3661	Event number 4, 5, 6, 11 (connect, delete, disconnect, unbind)
LDAP_MAPCERT_OPT	CHAR	2	3664	3665	Value is always 00
LDAP_MAPPED_SAFID	CHAR	8	3668	3675	The SAF ID, if any, associated with the bind DN. Not used for event codes 4 and 6.
LDAP_POLICY_UPDATED	CHAR	1	3678	3678	Blank for this operation.
LDAP_FLAGS	CHAR	10	3681	3690	Character representation of the hex value for flag settings. See note 2 on page 724 for more information.
LDAP_RESERVED_04	CHAR	227	3693	3919	Reserved
LDAP_PROTOCOL_VER	CHAR	2	3922	3923	The LDAP protocol version set to **2** or **3**.
LDAP_RETURN_CODE	INT	10	3926	3935	The LDAP server return code in decimal (blank if there was no return code provided).
LDAP_ERROR_MSG	CHAR	256	3938	4193	The reason code message number and text, if there is one, sent from the server to the client.
LDAP_ENTRY_NM	CHAR	512	4196	4707	The target entry of the requested operation (bind DN, DN to be deleted, DN to be added, and so on).
LDAP_RELOC_REQ_TS	CHAR	32	4710	4741	The time and date the request was received in the following format: `Tue Oct 13 16:36:25 2009`
LDAP_ENTRY_SUFFIX	CHAR	64	4744	4807	The normalized DN of the backend suffix that handled the request.
LDAP_CNTRLS_PRESNT	CHAR	512	4810	5321	The control OID and the criticality of the control in the request. Each control in the request has its own relocate. It has the following format: `ControlOID,TRUE|FALSE`

Table 97. Event specific fields for LDAP extended operations event (Event code 7)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_RESERVED_01	CHAR	16	3000	3015	Reserved
LDAP_REQ_TIMESTP	CHAR	16	3018	3033	The elapsed time for request completion (in microseconds).
LDAP_SERVER_URL	CHAR	32	3036	3067	The server's IPv4 or IPv6 address and port number (in LDAP URL format) that processed the client connection.
LDAP_CONN_ID	CHAR	8	3070	3077	The internal connection ID used to indicate operations performed on the same client connection.
LDAP_MESSAGE_ID	CHAR	8	3080	3087	The message ID from the BER encoded data from the client which is used to connect events.
LDAP_BIND_DN	CHAR	512	3090	3601	The bind DN for the connection.

Table 97. Event specific fields for LDAP extended operations event (Event code 7) (continued)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_CLIENT_SECL	CHAR	32	3604	3635	The LDAP client security label, if there is one.
LDAP_SRC_IP_ADDR	CHAR	16	3638	3653	The IP address of the client performing the request. If set to **PC**, the request came across the PC interface. It is set to **XCF** for sysplex replica requests and to **SLAPI** for internal SLAPI requests.
LDAP_AUDIT_VERSION	CHAR	2	3656	3657	The version of the LDAP audit support in use. It is always set to **V1**.
LDAP_EVENT_CODE	CHAR	2	3660	3661	Event number 7 (extended operations)
LDAP_MAPCERT_OPT	CHAR	2	3664	3665	Value is always 00
LDAP_MAPPED_SAFID	CHAR	8	3668	3675	The SAF ID, if any, associated with the bind DN.
LDAP_POLICY_UPDATED	CHAR	1	3678	3678	Blank for this operation.
LDAP_FLAGS	CHAR	10	3681	3690	Character representation of the hex value for flag settings. See note 2 on page 724 for more information.
LDAP_RESERVED_04	CHAR	227	3693	3919	Reserved
LDAP_PROTOCOL_VER	CHAR	2	3922	3923	The LDAP protocol version set to **2** or **3**.
LDAP_RETURN_CODE	INT	10	3926	3935	The LDAP server return code in decimal (blank if there was no return code provided).
LDAP_ERROR_MSG	CHAR	256	3938	4193	The reason code message number and text, if there is one, sent from the server to the client.
LDAP_ENTRY_NM	CHAR	512	4196	4707	The target entry of the requested operation (bind DN, DN to be deleted, DN to be added, and so on).
LDAP_RELOC_REQ_TS	CHAR	32	4710	4741	The time and date the request was received in the following format: `Tue Oct 13 16:36:25 2009`
LDAP_ENTRY_SUFFIX	CHAR	64	4744	4807	The normalized DN of the backend suffix that handled the request.
LDAP_CNTRLS_PRESNT	CHAR	512	4810	5321	The control OID and the criticality of the control in the request. Each control in the request has its own relocate. It has the following format: `ControlOID,TRUE|FALSE`
LDAP_XOP_REQ_NAME	CHAR	64	5324	5387	The OID of the extended operation request.

Table 98. Event specific fields for LDAP modify event (Event code 8)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_RESERVED_01	CHAR	16	3000	3015	Reserved
LDAP_REQ_TIMESTP	CHAR	16	3018	3033	The elapsed time for request completion (in microseconds).
LDAP_SERVER_URL	CHAR	32	3036	3067	The server's IPv4 or IPv6 address and port number (in LDAP URL format) that processed the client connection.
LDAP_CONN_ID	CHAR	8	3070	3077	The internal connection ID used to indicate operations performed on the same client connection.
LDAP_MESSAGE_ID	CHAR	8	3080	3087	The message ID from the BER encoded data from the client which is used to connect events.
LDAP_BIND_DN	CHAR	512	3090	3601	The bind DN for the connection.
LDAP_CLIENT_SECL	CHAR	32	3604	3635	The LDAP client security label, if there is one.
LDAP_SRC_IP_ADDR	CHAR	16	3638	3653	The IP address of the client performing the request. If set to **PC**, the request came across the PC interface. It is set to **XCF** for sysplex replica requests and to **SLAPI** for internal SLAPI requests.

Table 98. Event specific fields for LDAP modify event (Event code 8) (continued)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_AUDIT_VERSION	CHAR	2	3656	3657	The version of the LDAP audit support in use. It is always set to **V1**.
LDAP_EVENT_CODE	CHAR	2	3660	3661	Event number 8 (modify)
LDAP_MAPCERT_OPT	CHAR	2	3664	3665	Value is always 00
LDAP_MAPPED_SAFID	CHAR	8	3668	3675	The SAF ID, if any, associated with the bind DN.
LDAP_POLICY_UPDATED	CHAR	1	3678	3678	Blank for this operation.
LDAP_FLAGS	CHAR	10	3681	3690	Character representation of the hex value for flag settings. See note 2 on page 724 for more information.
LDAP_RESERVED_04	CHAR	227	3693	3919	Reserved
LDAP_PROTOCOL_VER	CHAR	2	3922	3923	The LDAP protocol version set to **2** or **3**.
LDAP_RETURN_CODE	INT	10	3926	3935	The LDAP server return code in decimal (blank if there was no return code provided).
LDAP_ERROR_MSG	CHAR	256	3938	4193	The reason code message number and text, if there is one, sent from the server to the client.
LDAP_ENTRY_NM	CHAR	512	4196	4707	The target entry of the requested operation (bind DN, DN to be deleted, DN to be added, and so on).
LDAP_RELOC_REQ_TS	CHAR	32	4710	4741	The time and date the request was received in the following format: `Tue Oct 13 16:36:25 2009`
LDAP_ENTRY_SUFFIX	CHAR	64	4744	4807	The normalized DN of the backend suffix that handled the request.
LDAP_CNTRLS_PRESNT	CHAR	512	4810	5321	The control OID and the criticality of the control in the request. Each control in the request has its own relocate. It has the following format: `ControlOID,TRUE|FALSE`
LDAP_MOD_ATTR_DEL	CHAR	64 (maximum 909)	5324	6233	The name of the attribute deleted in the modify request. Each attribute type deleted in the modify request has its own relocate and is separated by 1 space. There is a maximum length of 909 bytes available for all attribute types deleted in the modify request.
LDAP_MOD_ATTR_ADD	CHAR	64 (maximum 909)	6236	7145	The name of the attribute added in the modify request. Each attribute type added in the modify request has its own relocate and is separated by 1 space. There is a maximum length of 909 bytes available for all attribute types added in the modify request.
LDAP_MOD_ATTR_REP	CHAR	64 (maximum 909)	7148	8057	The name of the attribute replaced in the modify request. Each attribute type replaced in the modify request has its own relocate and is separated by 1 space. There is a maximum length of 909 bytes available for all attribute types replaced in the modify request.

Table 99. Event specific fields for LDAP modify DN event (Event code 9)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_RESERVED_01	CHAR	16	3000	3015	Reserved
LDAP_REQ_TIMESTP	CHAR	16	3018	3033	The elapsed time for request completion (in microseconds).
LDAP_SERVER_URL	CHAR	32	3036	3067	The server's IPv4 or IPv6 address and port number (in LDAP URL format) that processed the client connection.
LDAP_CONN_ID	CHAR	8	3070	3077	The internal connection ID used to indicate operations performed on the same client connection.

Table 99. Event specific fields for LDAP modify DN event (Event code 9) (continued)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_MESSAGE_ID	CHAR	8	3080	3087	The message ID from the BER encoded data from the client which is used to connect events.
LDAP_BIND_DN	CHAR	512	3090	3601	The bind DN for the connection.
LDAP_CLIENT_SECL	CHAR	32	3604	3635	The LDAP client security label, if there is one.
LDAP_SRC_IP_ADDR	CHAR	16	3638	3653	The IP address of the client performing the request. If set to **PC**, the request came across the PC interface. It is set to **XCF** for sysplex replica requests and to **SLAPI** for internal SLAPI requests.
LDAP_AUDIT_VERSION	CHAR	2	3656	3657	The version of the LDAP audit support in use. It is always set to **V1**.
LDAP_EVENT_CODE	CHAR	2	3660	3661	Event number 9 (modify DN)
LDAP_MAPCERT_OPT	CHAR	2	3664	3665	Value is always 00
LDAP_MAPPED_SAFID	CHAR	8	3668	3675	The SAF ID, if any, associated with the bind DN.
LDAP_POLICY_UPDATED	CHAR	1	3678	3678	Blank for this operation.
LDAP_FLAGS	CHAR	10	3681	3690	Character representation of the hex value for flag settings. See note 2 on page 724 for more information.
LDAP_RESERVED_04	CHAR	227	3693	3919	Reserved
LDAP_PROTOCOL_VER	CHAR	2	3922	3923	The LDAP protocol version set to **2** or **3**.
LDAP_RETURN_CODE	INT	10	3926	3935	The LDAP server return code in decimal (blank if there was no return code provided).
LDAP_ERROR_MSG	CHAR	256	3938	4193	The reason code message number and text, if there is one, sent from the server to the client.
LDAP_ENTRY_NM	CHAR	512	4196	4707	The target entry of the requested operation (bind DN, DN to be deleted, DN to be added, and so on).
LDAP_RELOC_REQ_TS	CHAR	32	4710	4741	The time and date the request was received in the following format: `Tue Oct 13 16:36:25 2009`
LDAP_ENTRY_SUFFIX	CHAR	64	4744	4807	The normalized DN of the backend suffix that handled the request.
LDAP_CNTRLS_PRESNT	CHAR	512	4810	5321	The control OID and the criticality of the control in the request. Each control in the request has its own relocate. It has the following format: `ControlOID,TRUE|FALSE`
LDAP_MDN_NEW_RDN	CHAR	32	5324	5355	The new relative distinguished name for the target entry in the **modifydn** request.
LDAP_MDN_DOLD_RDN	CHAR	1	5358	5358	**T** (true) or **F** (false) if the old relative distinguished name (RDN) is deleted in the **modifydn** request.
LDAP_MDN_NEW_SUP	CHAR	512	5361	5872	The new superior, if there is one, of the target entry specified in the **modifydn** request.

Table 100. Event specific fields for LDAP search event (Event code 10)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_RESERVED_01	CHAR	16	3000	3015	Reserved
LDAP_REQ_TIMESTP	CHAR	16	3018	3033	The elapsed time for request completion (in microseconds).
LDAP_SERVER_URL	CHAR	32	3036	3067	The server's IPv4 or IPv6 address and port number (in LDAP URL format) that processed the client connection.
LDAP_CONN_ID	CHAR	8	3070	3077	The internal connection ID used to indicate operations performed on the same client connection.

Table 100. Event specific fields for LDAP search event (Event code 10) (continued)

Field name	Type	Length	Position		Comments
			Start	End	
LDAP_MESSAGE_ID	CHAR	8	3080	3087	The message ID from the BER encoded data from the client which is used to connect events.
LDAP_BIND_DN	CHAR	512	3090	3601	The bind DN for the connection.
LDAP_CLIENT_SECL	CHAR	32	3604	3635	The LDAP client security label, if there is one.
LDAP_SRC_IP_ADDR	CHAR	16	3638	3653	The IP address of the client performing the request. If set to **PC**, the request came across the PC interface. It is set to **XCF** for sysplex replica requests and to **SLAPI** for internal SLAPI requests.
LDAP_AUDIT_VERSION	CHAR	2	3656	3657	The version of the LDAP audit support in use. It is always set to **V1**.
LDAP_EVENT_CODE	CHAR	2	3660	3661	Event number 10 (search)
LDAP_MAPCERT_OPT	CHAR	2	3664	3665	Value is always 00
LDAP_MAPPED_SAFID	CHAR	8	3668	3675	The SAF ID, if any, associated with the bind DN.
LDAP_POLICY_UPDATED	CHAR	1	3678	3678	Blank for this operation.
LDAP_FLAGS	CHAR	10	3681	3690	Character representation of the hex value for flag settings. See note 2 on page 724 for more information.
LDAP_RESERVED_04	CHAR	227	3693	3919	Reserved
LDAP_PROTOCOL_VER	CHAR	2	3922	3923	The LDAP protocol version set to **2** or **3**.
LDAP_RETURN_CODE	INT	10	3926	3935	The LDAP server return code in decimal (blank if there was no return code provided).
LDAP_ERROR_MSG	CHAR	256	3938	4193	The reason code message number and text, if there is one, sent from the server to the client.
LDAP_ENTRY_NM	CHAR	512	4196	4707	The target entry of the requested operation (bind DN, DN to be deleted, DN to be added, and so on).
LDAP_RELOC_REQ_TS	CHAR	32	4710	4741	The time and date the request was received in the following format: `Tue Oct 13 16:36:25 2009`
LDAP_ENTRY_SUFFIX	CHAR	64	4744	4807	The normalized DN of the backend suffix that handled the request.
LDAP_CNTRLS_PRESNT	CHAR	512	4810	5321	The control OID and the criticality of the control in the request. Each control in the request has its own relocate. It has the following format: `ControlOID,TRUE|FALSE`
LDAP_SEARCH_FILTER	CHAR	256	5324	5579	The search filter specified on the search request.
LDAP_SEARCH_SCOPE	CHAR	12	5582	5593	The scope of the search request. It is set to **SUBTREE** for subtree level searches, **ONE** for one level searches, and **BASE** for base level searches.
LDAP_SEARCH_ATTRS	CHAR	64	5596	5659	The name of the attribute requested to be returned on the search request. Each attribute type requested in the search request has its own relocate and is separated by 1 space. There is a maximum length of 833 bytes available for all requested attribute types in the search request.

Appendix F. Activity log records

This section describes the fields that are present in the activity log when it is configured to log z/OS LDAP client requests.

Note: The activity log is not an official interface to the LDAP server. It may change at any time.

Activity log start and end field descriptions

Table 101 describes the fields that are present in start or end activity log records. Activity log start records are logged when the **GLDLOG_TIME** environment variable is set to **notime**, or **logFileRecordType** is set to begin in the configuration file. Activity log start and end records are logged when the **GLDLOG_TIME** environment variable is set to **time** or **logFileRecordType** is set to both in the configuration file.

Table 101. Start or end activity log fields

Field name	Operations	Description
attrs	search, add(V1+), modify(V1+)	The name of the attributes that are requested to be returned on the search request.
base	search	The base DN for the search request.

Table 101. Start or end activity log fields (continued)

Field name	Operations	Description

bindFlags	bind	The hex value for the following possible flag settings is:
		`ADMIN_ROLE_NONE 0x00000001`
		`ADMIN_ROLE_ROOT 0x00000002`
		`ADMIN_ROLE_DIR 0x00000004`
		`ADMIN_ROLE_REPL 0x00000008`
		`ADMIN_ROLE_SCHEMA 0x00000010`
		`ADMIN_ROLE_CONFIG 0x00000020`
		`ADMIN_ROLE_PASSWORD 0x00000040`
		`ADMIN_ROLE_OPER 0x00000080`
		`ADMIN_ROLE_SAF 0x00000100`
		where, the bound user has one or more of the following administrative roles,
		• ADMIN_ROLE_NONE=No administrator
		• ADMIN_ROLE_ROOT=Root administrator
		• ADMIN_ROLE_DIR=Directory data administrator
		• ADMIN_ROLE_REPL=Replication administrator
		• ADMIN_ROLE_SCHEMA=Schema administrator
		• ADMIN_ROLE_CONFIG=Server configuration group member
		• ADMIN_ROLE_PASSWORD=Password administrator
		• ADMIN_ROLE_OPER=Operational administrator
		• ADMIN_ROLE_SAF=The administrative user is a member of **cn=safadmingroup,cn=configuration** and the roles are defined in RACF.
		Example: If the user is a member of **cn=safadmingroup,cn=configuration** and has been permitted to all administrative roles, `bindFlags = 1FF` is in this field.

cause	disconnect(V1+)	The reason code value for disconnect, the possible values, and description are: 1: Idle connection time-out 2: Network interface failed. 4: Connection closed abnormally. 8: Client closed connection. 10: The request is not a valid LDAP message. 20: Internal error. For example: out of storage error, unable to encode or decode LDAP message. 40: Failed to write LDAP message to network connection. 80: Error occurred in System SSL. For example, if the value is 5, it means that the disconnect cause is: (1) Idle connection time-out and (2) Connection closed abnormally.

Table 101. Start or end activity log fields (continued)

Field name	Operations	Description
connid	add, bind, compare, delete, extendedop, modify, modrdn, search, unbind, connect(V1+), disconnect(V1+), abandon(V1+), unknown(V1+)	The internal connection ID used to indicate operations performed.
count	search	The number of entries that are returned on the search request.
DN	add, bind, delete, disconnect(V1+), modify, modrdn, unbind	The target entry of the operation (base DN for the search, bind DN, DN to be deleted, DN to be added, and so on).
entry	compare	The target entry of the compare operation.
filter	search	The search filter that is specified in the search request.
IP	add, bind, compare, delete, extendedop, modify, modrdn, search, unbind, connect(V1+), disconnect(V1+), abandon(V1+), unknown(V1+)	The IP address of the client performing the request.
listen	bind, unbind, connect(V1+)	The listen URL that the LDAP server is listening on that received the client connection.
msgid	add(V1+), bind(V1+), compare(V1+), delete(V1+), exop(V1+), modify(V1+), search(V1+), abandon(V1+), unknown(V1+)	Message Identifier.
newRdn	modrdn	The new relative distinguished name for the target entry in the modrdn request.
policyUpdated	bind	• **T** (true) or **F** (false) if the password policy operational attribute values in the bind DN entry are updated. • **NA** if the operation is compare and the attribute that is being compared is not **userPassword**.
pReqOID	extendedop	The OID of the extended operation request.
rc	add, bind, compare, delete, extendedop, modify, modrdn, search, unbind	The LDAP server return code for the operation.
safid	bind	The SAF ID, if any, associated with the bind DN.
scope	search	The scope of the search request. It is set to 0 for subtree searches, 1 for one level searches, and 2 for subtree searches.

Table 101. Start or end activity log fields (continued)

Field name	Operations	Description

searchFlags	search	The hex value for the following possible flag settings is:
		`SEARCH_PAGE_FIRST 0x00000001`
		`SEARCH_PAGE_NEXT 0x00000002`
		`SEARCH_PAGE_LAST 0x00000004`
		`SEARCH_SORT_COMPLETE 0x00000008`
		
		where,
		• SEARCH_PAGE_FIRST=First page of a paged search request
		• SEARCH_PAGE_NEXT=Ensuing page of a paged search request
		• SEARCH_PAGE_LAST=Last page of a paged search request
		• SEARCH_SORT_COMPLETE=Sorted search requested and completed successfully
		
		Example: If the last page of a successfully paged and sorted search request has been returned to the client application, `searchFlags = 0E` is set in this field in the search end activity log.

targetmsgid	abandon(V1+)	The target message ID of the abandon operation.
type	unknown(V1+)	The type of the request.

Note: Fields for the **logFileVersion** 1 or above are indicated with (V1+).

Activity log mergedRecord field descriptions

Table 102 describes the fields that are present in mergedRecord activity log records. Activity log mergedRecords are logged when the **GLDLOG_TIME** environment variable is set to **mergedrecord** or **logFileRecordType** is set to **mergedRecord** in the configuration file.

Table 102. mergedRecord activity log fields

Field name	Operations	Description
attr	compare	The name of the attribute being compared.
attrs	add(V1+), modify(V1+), search	The name of the attributes that are requested to be returned on the request.
bind	add, bind, compare, disconnect(V1+), delete, extendedop, modify, modrdn, search, unbind, unknown(V1+)	The bind DN for the connection.

Table 102. mergedRecord activity log fields (continued)

Field name	Operations	Description

bindFlags	bind	The hex value for the following possible flag settings is:
		`ADMIN_ROLE_NONE 0x00000001`
		`ADMIN_ROLE_ROOT 0x00000002`
		`ADMIN_ROLE_DIR 0x00000004`
		`ADMIN_ROLE_REPL 0x00000008`
		`ADMIN_ROLE_SCHEMA 0x00000010`
		`ADMIN_ROLE_CONFIG 0x00000020`
		`ADMIN_ROLE_PASSWORD 0x00000040`
		`ADMIN_ROLE_OPER 0x00000080`
		`ADMIN_ROLE_SAF 0x00000100`
		where,
		• ADMIN_ROLE_NONE=No administrator
		• ADMIN_ROLE_ROOT=Root administrator
		• ADMIN_ROLE_DIR=Directory data administrator
		• ADMIN_ROLE_REPL=Replication administrator
		• ADMIN_ROLE_SCHEMA=Schema administrator
		• ADMIN_ROLE_CONFIG=Server configuration group member
		• ADMIN_ROLE_PASSWORD=Password administrator
		• ADMIN_ROLE_OPER=Operational administrator
		• ADMIN_ROLE_SAF=The administrative user is a member of **cn=safadmingroup,cn=configuration** and the roles are defined in RACF.
		Example: If the user is a member of **cn=safadmingroup,cn=configuration** and has been permitted to all administrative roles, `bindFlags = 1FF` is in this field.

cause	disconnect(V1+)	The reason code value for disconnect, the possible values, and description are: 1: Idle connection time-out 2: Network interface failed. 4: Connection closed abnormally. 8: Client closed connection. 10: The request is not a valid LDAP message. 20: Internal error. For example: out of storage error, unable to encode or decode LDAP message. 40: Failed to write LDAP message to network connection. 80: Error occurred in System SSL. For example, if the value is 5, it means that the disconnect cause is: (1) Idle connection time-out and (2) Connection closed abnormally.

Table 102. mergedRecord activity log fields (continued)

Field name	Operations	Description
clientIP	abandon(V1+), add, bind, compare, connect(V1+), delete, disconnect(V1+), extendedop, modify, modrdn, search, unbind, unknown(V1+)	The IP address of the client performing the request.
connid	abandon(V1+), add, bind, compare, connect(V1+), delete, disconnect(V1+), extendedop, modify, modrdn, search, unbind, unknown(V1+)	The internal connection ID used to indicate operations performed on the same client connection.
controls	add, bind, compare, delete, extendedop, modify, modrdn, search, unbind, unknown(V1+)	The OID of the control on the request. If there are multiple controls on the request, they are separated by ':'.
count	search	The number of entries that are returned on the search request.
delete oldRdn	modrdn	**T** (true) or **F** (false) if the old relative distinguished name (RDN) is deleted in the modrdn request.
filter	search	The search filter that is specified in the search request.
listen	bind, connect(V1+), unbind	The listen URL that the LDAP server was listening on that received the client connection.
mech	bind	Indicates the authentication method. It has one of the following values: **SIMPLE, EXTERNAL, GSSAPI, DIGEST-MD5**, or **CRAM-MD5**
msgid	abandon(V1+), add(V1+), bind(V1+), compare(V1+), delete(V1+), exop(V1+), modify(V1+), search(V1+), unknown(V1+)	Message identifier.
newRdn	modrdn	The new relative distinguished name for the target entry in the modrdn request.
policyUpdated	bind	• **T** (true) or **F** (false) if the password policy operational attribute values in the bind DN entry are updated. • **NA** if the operation is compare and the attribute that is being compared is not **userPassword**.
rc	add, bind, compare, delete, extendedop, modify, modrdn, search, unbind, unknown(V1+)	The LDAP server return code for the operation.
rsn	add, bind, compare, delete, extendedop, modify, modrdn, search, unbind, unknown(V1+)	The LDAP reason code message number and text, if there is one, sent from the server to the client.
saf	bind, unbind	The SAF ID, if any, associated with the bind DN.
scope	search	The scope of the search request. It is set to 0 for subtree searches, 1 for one level searches, and 2 for subtree searches.

Table 102. mergedRecord activity log fields (continued)

Field name	Operations	Description

searchFlags	search	The hex value for the following possible flag settings is:
		`SEARCH_PAGE_FIRST 0x00000001`
		`SEARCH_PAGE_NEXT 0x00000002`
		`SEARCH_PAGE_LAST 0x00000004`
		`SEARCH_SORT_COMPLETE 0x00000008`
		
		where,
		• SEARCH_PAGE_FIRST=First page of a paged search request
		• SEARCH_PAGE_NEXT=Ensuing page of a paged search request
		• SEARCH_PAGE_LAST=Last page of a paged search request
		• SEARCH_SORT_COMPLETE=Sorted search requested and completed successfully
		
		Example: If the last page of a successfully paged and sorted search request has been returned to the client application, `searchFlags = 0E` is set in this field.

seclabel	bind, unbind	The LDAP client security label, if there is one.
target	add, compare, modify, modrdn, search	The target entry of the operation (base DN for the search, bind DN, DN to be deleted, DN to be added, and so on).
targetmsgid	abandon(V1+)	The target message ID of the abandon operation.
time	abandon(V1+), add, bind, compare, delete, extendedop, modify, modrdn, search, unbind, unknown(V1+)	The elapsed time for request completion (in microseconds).
type	unknown(V1+)	The type of the request.
XOP	extendedop	The OID of the extended operation request.

Note: Fields for the **logFileVersion** 1 or above are indicated with (V1+).

Appendix G. Guidelines for interoperability between non-z/OS TDS and z/OS TDS

This section contains information to consider when setting up a mixed platform environment where an LDAP directory is being replicated between non-z/OS TDS on Linux, UNIX, or Windows platforms and z/OS TDS. This information also applies to migration of the schema and directory entries between these different platforms.

Schema considerations

Syntax and matching rules

There are some minor differences in the support of some syntaxes and matching rules between z/OS TDS and non-z/OS TDS.

•
  - z/OS TDS checks the characters used in a value of an attribute with Facsimile Telephone Number syntax, Printable String syntax, Telex Number syntax, or Telephone Number syntax. Non-z/OS TDS does not check values for these syntaxes.
  - z/OS TDS supports specifying a space-separated list of numbers for a value of an attribute with Numeric String syntax. Non-z/OS TDS supports only a single number, with no spaces.
  - z/OS TDS allows any printable character in a value of an attribute with Country String syntax. Non-z/OS TDS allows only alphabetic characters. z/OS TDS also allows specifying the case ignore matching rule in an attribute with this syntax. Non-z/OS TDS does not support any matching rules (and always does a case ignore match).
  - z/OS TDS supports specifying `certficateMatch` and `certficateExactMatch` in an attribute with Certificate syntax. Non-z/OS TDS does not support these matching rules.

If you plan to migrate entries using attributes with these syntaxes, make sure to define the attributes using matching rules supported on both z/OS TDS and non-z/OS TDS, and restrict the attribute values to characters and formats that are supported by both. An attribute value that is not supported must be changed before the entry can be migrated.

Schema LDIF format

The format of the schema LDIF obtained from non-z/OS TDS or z/OS TDS by publishing the schema (using **ldapsearch –L**) might not be acceptable input for a schema modification.

- When modifying the non-z/OS TDS schema, break up the **attributetypes** and **objectclasses** in the schema file into separate schema modifications, each including a single **attributetypes** value or **objectclasses** value. Also, include an **ibmattributetypes** value (if any) in the modification for its associated **attributetypes** value. If the attribute or object class exists in the schema, the modification is modify-replace; otherwise, the modification is modify-add.

When modifying the z/OS TDS schema, the entire LDIF can be processed in a single modify-replace operation, whether the attributes or object classes exist in the schema.

For example, assume that attribute **attr1** and object class **objclass1** exist in the schema. For z/OS TDS, the following schema modification replaces those schema elements and adds new attribute **attr2** and object class **objclass2**:

```
dn: cn=schema
changetype: modify
replace: attributetypes
attributetypes: (
  1.3.18.0.2.4.11111
  NAME 'attr1'
  DESC 'Description for attribute attr1'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  USAGE userApplications
  )
IBMAttributetypes: (
  1.3.18.0.2.4.11111
  ACCESS-CLASS normal
  )
attributetypes: (
  1.3.18.0.2.4.22222
  NAME 'attr2'
  DESC 'Description for attribute attr2'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  USAGE userApplications
  )
IBMAttributetypes: (
  1.3.18.0.2.4.22222
  ACCESS-CLASS normal
  )
-
replace: objectclasses
objectclasses: (
  1.3.18.0.2.6.33333
  NAME 'objclass1'
  DESC 'Description for object class objclass1'
  SUP top
  STRUCTURAL
  MUST ( cn )
  MAY ( attr1 )
  )
objectclasses: (
  1.3.18.0.2.6.44444
  NAME 'objclass2'
  DESC 'Description for object class objclass2'
  SUP top
  STRUCTURAL
  MUST ( cn )
  MAY ( attr1 $ attr2 )
  )
```

For non-z/OS TDS, this schema modification must be reformatted into separate schema modifications and modify-add used instead of modify-replace for the new schema elements, as follows:

```
dn: cn=schema
changetype: modify
replace: attributetypes
attributetypes: (
  1.3.18.0.2.4.11111
  NAME 'attr1'
  DESC 'Description for attribute attr1'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  USAGE userApplications
  )
```

```
IBMAttributetypes: (
  1.3.18.0.2.4.11111
  ACCESS-CLASS normal
  )

dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: (
  1.3.18.0.2.4.22222
  NAME 'attr2'
  DESC 'Description for attribute attr2'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  USAGE userApplications
  )
IBMAttributetypes: (
  1.3.18.0.2.4.22222
  ACCESS-CLASS normal
  )

dn: cn=schema
changetype: modify
replace: objectclasses
objectclasses: (
  1.3.18.0.2.6.33333
  NAME 'objclass1'
  DESC 'Description for object class objclass1'
  SUP top
  STRUCTURAL
  MUST ( cn )
  MAY ( attr1 )
  )

dn: cn=schema
changetype: modify
add: objectclasses
objectclasses: (
  1.3.18.0.2.6.44444
  NAME 'objclass2'
  DESC 'Description for object class objclass2'
  SUP top
  STRUCTURAL
  MUST ( cn )
  MAY ( attr1 $ attr2 )
  )
```

- Ensure that the object classes do not precede the attributes that they refer to.

Import or export of directory entries

Exporting data from non-z/OS TDS to z/OS TDS

- Non-z/OS TDS includes certain suffixes, such as **cn=configuration, cn=ibmPolicies**, and **cn=localhost**, that contain special entries used to manage LDAP configuration, policies, and replication. z/OS TDS only supports some of these special entries. See "CDBM backend configuration and policy entries" on page 139 for information about the supported special entries in z/OS TDS. Remove the other non-z/OS TDS special entries from the LDIF or use **db2ldif -s *subtreeDN* -x** to avoid unloading these suffixes.
- User passwords must be in clear text, crypt, SHA, Salted SHA (SSHA), SHA224, SSHA224 (Salted SHA224), SHA256, SSHA256 (Salted SHA256), SHA384, SSHA384 (Salted SHA384), SHA512, or SSHA512 (Salted SHA512). Other forms are not compatible with z/OS TDS. If using crypt, make certain to specify **pwCryptCompat off** in the z/OS TDS configuration file.

- z/OS TDS does not support the use of filtered ACLs in the **ibm-filterAclEntry** attribute. You must remove these before importing to z/OS TDS.

Exporting data from z/OS TDS to non-z/OS TDS

- For non-z/OS TDS, **aclEntry** and **entryOwner** attribute values must begin with the following format:

  ```
  "access-id:|group:|role:"
  ```

  This is not required for z/OS TDS, therefore, it might need to be added to these attribute values before importing to non-z/OS TDS. Always specify these on z/OS TDS to avoid this issue.
- User passwords must be in clear text, crypt, SHA, Salted SHA (SSHA), SHA224, SSHA224 (Salted SHA224), SHA256, SSHA256 (Salted SHA256), SHA384, SSHA384 (Salted SHA384), SHA512, or SSHA512 (Salted SHA512). Other forms are not compatible with non-z/OS TDS. If using crypt, make certain to specify **pwCryptCompat off** in the z/OS TDS configuration file. Use the **-t** option of the **ds2ldif** utility to unload passwords in the tagged format used by non-z/OS TDS. If importing SHA-2 or Salted SHA-2 user passwords into non-z/OS TDS 6.3 or greater, the **ibm-slapdUseNonFIPSCrypt** attribute in the **cn=configuration** entry must be set to true before doing the importing.
- Non-z/OS TDS does not support ACL filters in the **aclEntry** and **entryOwner** attribute values. If there are **aclEntry** or **entryOwner** attribute values that have **aclFilter** or **ownerFilter** scopes of protection, they must be removed before importing to non-z/OS TDS.

Functional considerations

- For non-z/OS TDS, deleting a person entry results in removing the DN of the entry from groups and ACLs. This is not done in z/OS TDS. Instead, applications must do this themselves.
- Similarly, non-z/OS TDS supports a control that allows deletion of all entries in a subtree. z/OS TDS does not support this control, therefore, applications must do this themselves.
- There are some capabilities that only non-z/OS TDS or only z/OS TDS supports. Restrict usage to capabilities supported on both platforms to facilitate replication of operations and migration of entries.
- For non-z/OS TDS, group search limits are only used when they are contained in group entries located under `cn=localhost` or `cn=ibmpolicies`. Furthermore, the limits in groups under `cn=ibmpolicies` are ignored if there are limits in groups under `cn=localhost`. In z/OS TDS, group entries containing group search limits can be located anywhere in LDBM, CDBM, and TDBM backends and the limits in all groups are used.
- For non-z/OS TDS, the underlying DB2 database is taken advantage of to perform sorting of search results, resulting in potentially different sorted search results based on the data code page for the database. z/OS TDS does not have this restriction.
- For non-z/OS TDS, sorting of search results is limited to examination of the first 240 bytes based on its use of DB2 sorting on the index and index length limitations. z/OS TDS does not have this limitation.

Administrative group and roles considerations

- Dynamic configuration is supported for the administrative group and roles in z/OS TDS. Administrative group member entries and roles are defined under the **cn=admingroup,cn=configuration** entries or in the **cn=safadmingroup,cn=configuration** entry in the CDBM backend.
- Only simple binds are supported for administrative group members defined in the CDBM backend. However, more complex bind mechanisms, such as CRAM-MD5 and DIGEST-MD5 binds, can be used by pointing to an entry in the LDBM or TDBM backend. Then, all bind mechanisms are supported.
- If an administrative group member is in the DIT and it exists in an advanced replication context or backend (basic replication), the entry is replicated to other servers. If the other consumer or replica servers do not have the administrative group and roles set up correctly or ACLs are not set correctly, the administrative group member's credentials can be exposed.
- Administrative group members might be subject to ACL filters.
- The Audit log administrator and the Start/Stop administrator roles are not supported in z/OS TDS. However, z/OS TDS provides two additional administrative roles, root administrator and operational administrator.

Appendix H. Searching operational attributes

The LDAP server supports many operational attributes that can be returned on search operations. However, many are derived and are not supported within search filters. If a non-searchable operational attribute is included on a search filter, it typically does not produce an error. Instead, a FALSE match is produced, making its use in a search filter predicate ineffective. Table 103 lists operational attributes that can be returned from a search and whether the attribute can be used effectively in a search filter predicate.

Table 103. Search operational attributes

Attribute name	User modifiable	Filter predicate	Description
aclEntry		Yes[1]	Defines an access list entry.
aclPropagate		Yes[1]	Defines access list subtree propagation.
aclSource	NO-USER-MODIFICATION	No	Source of the access list for an entry.
attributeTypes		No	LDAP attribute types.
createTimestamp	NO-USER-MODIFICATION	Yes	Entry creation time.
creatorsName	NO-USER-MODIFICATION	Yes	Name of entry creator.
directoryOperationName		No	The distinguished name of a **directoryOperation** object within the same organization.
directoryOperationString		No	A constructed directory operation request string.
ditContentRules		No	Directory content rules.
ditStructureRules		No	Directory structure rules.
entryOwner		Yes[1]	Defines an entry owner.
hasSubordinates	NO-USER-MODIFICATION	No	Indicates whether any subordinate entries exist below the entry holding this attribute.
ibm-allGroups	NO-USER-MODIFICATION	No	Lists all groups containing an entry.
ibm-allMembers	NO-USER-MODIFICATION	No	Lists all members of a group.
ibm-effectiveReplicationModel	NO-USER-MODIFICATION	No	Advertises in the root DSE the OID of the replication model in use by the server.
ibm-entryChecksum	NO-USER-MODIFICATION	No	A checksum of the user attributes for the entry containing this attribute.
ibm-entryChecksumOp	NO-USER-MODIFICATION	No	A checksum of the replicated operational attributes for the entry containing this attribute.
ibm-EntryUUID	NO-USER-MODIFICATION	Yes	Uniquely identifies an LDAP entry throughout its life.
ibm-pwdAccountLocked		Yes	The indication that the users account is locked.

Table 103. Search operational attributes (continued)

Attribute name	User modifiable	Filter predicate	Description
ibm-pwdGroupPolicyDN		Yes	DN of an entry containing password policy information. This entry can be used in a group entry to associate a password policy with the entry.
ibm-pwdIndividualPolicyDN		Yes	DN of an entry containing password policy information. This entry can be used in a user entry to associate a password policy with the entry.
ibm-replicaPKCS11Enabled		Yes	Must be one of { TRUE | FALSE }. Specify whether PKCS11 interface is enabled to do cryptographic operation and key database file lookup from the installed Crypto device.
ibm-replicationChangeLDIF	NO-USER-MODIFICATION	No	Provides LDIF representation of the last failing operation.
ibm-replicationFailedChangeCount	NO-USER-MODIFICATION	No	Indicates the number of changes that are logged as failures for this replication agreement.
ibm-replicationFailedChanges	NO-USER-MODIFICATION	No	Unreplicated change for a specific replication agreement in the form <change id> <operation> <dn> <LDAP result code> <timestamp> <failed attempts> where operation is ADD, DELETE, MODIFY, or MODIFYDN.
ibm-replicationIsQuiesced		No	Indicates whether the replicated subtree containing this attribute is quiesced on this server.
ibm-replicationLastActivationTime	NO-USER-MODIFICATION	No	Indicates the last time the replication thread was activated.
ibm-replicationLastChangeId	NO-USER-MODIFICATION	No	Indicates last change ID successfully replicated for a replication agreement.
ibm-replicationLastFinishTime	NO-USER-MODIFICATION	No	Indicates the last time the replication thread completed sending all of the pending entries.
ibm-replicationLastResult	NO-USER-MODIFICATION	No	Result of last attempted replication in the form: <time> <change-id> <result code> <operation> <entry-dn>
ibm-replicationLastResultAdditional	NO-USER-MODIFICATION	No	Provides any additional error information returned by the consuming server in the message component of the LDAP result.
ibm-replicationNextTime	NO-USER-MODIFICATION	No	Indicates next scheduled time for replication.
ibm-replicationPendingChangeCount	NO-USER-MODIFICATION	No	Indicates the total number of pending unreplicated changes for this replication agreement.
ibm-replicationPendingChanges	NO-USER-MODIFICATION	No	Unreplicated change in the form <change id> <operation> <dn> where operation is ADD, DELETE, MODIFY, MODIFYDN.

Table 103. Search operational attributes (continued)

Attribute name	User modifiable	Filter predicate	Description
ibm-replicationperformance	NO-USER-MODIFICATION	No	Values for the following metrics: connection, operations queued, dependent operations queued, operations sent, dependent operations sent, operations received, errors.
ibm-replicationState	NO-USER-MODIFICATION	No	Indicates the state of the replication thread: active, ready, waiting, suspended, or full. If full, the value indicates the amount of progress.
ibm-replicationThisServerIsMaster	NO-USER-MODIFICATION	No	Indicates whether the server returning this attribute is a master server for the subtree containing this entry.
ibm-searchSizeLimit		Yes	Maximum number of entries to return from search requests for a member in a special search limit group. 0 = unlimited. -1 = ignored
ibm-searchTimeLimit		Yes	Maximum number of seconds to spend on search requests for a member in a special search limit group. 0 = unlimited. -1 = ignored
ibm-slapdAdminDN		Yes	Bind DN for **ibmslapd** administrator. (For example: `cn=root`)
ibm-slapdAdminGroupEnabled		Yes	Must be one of { TRUE | FALSE }. Specifies whether the administrative group is enabled. Defaults to FALSE if unspecified. If set to TRUE, the server allows users in the administrative group to log in.
ibm-slapdAdminPW		Yes	Bind password for **ibmslapd** administrator.
ibm-slapdAdminRole		Yes	Administrative roles associated with a local administrative group member. Valid values are **AuditAdmin, DirDataAdmin, NoAdmin, PasswordAdmin, ReplicationAdmin, SchemaAdmin, ServerConfigGroupMember, ServerStartStopAdmin**. Additional values that are supported by z/OS are **RootAdmin** and **OperationalAdmin**.
ibm-slapdDigestAdminUser		Yes	Specifies the DIGEST-MD5 user name of the LDAP administrator or administrative group member. Used when DIGEST-MD5 authentication is used to authenticate an administrator.
ibm-slapdDN	NO-USER-MODIFICATION	No	This attribute is used to sort search results by the entry DN.
ibm-slapdKrbAdminDN		Yes	Specifies the Kerberos ID of the LDAP administrator (For example: `ibm-kn=name@realm`). Used when Kerberos authentication is used to authenticate the administrator when logged on to the web administrator interface. This is specified instead of **adminDN** and **adminPW**.

Table 103. Search operational attributes (continued)

Attribute name	User modifiable	Filter predicate	Description
ibm-slapdMasterDN		Yes	Bind DN used by a replication supplier server. The value must match the **replicaBindDN** in the credentials object associated with the replication agreement. When Kerberos is used to authenticate to the replica, **ibm-slapdMasterDN** must specify the DN representation of the Kerberos ID (For example: `ibm-kn=freddy@realm1`). When Kerberos is used, **MasterServerPW** is ignored.
ibm-slapdMasterPW		Yes	Bind password used by replication supplier server. The value must match the **replicaBindPW** in the credentials object associated with the replication agreement. When Kerberos is used, **MasterServerPW** is ignored.
ibm-slapdMasterReferral		Yes	URL of master replica server. (For example: `ldaps://master.us.ibm.com:636`)
ibm-slapdMigrationInfo		Yes	Information that is used to control migration of a component.
ibm-slapdNoReplConflictResolution		Yes	Specifies whether directory server handles replication conflict resolution. If it is set to true, then the server does not try to compare time stamps for replicated entries in an attempt to resolve conflicts between the entries. However, conflict resolution does not apply to entry `cn=schema`, which is always replaced by a replicated `cn=schema`.
ibm-slapdPagedResAllowNonAdmin		Yes	Whether the server should allow non-Administrators to request paged search results.
ibm-slapdPagedResLmt		Yes	Maximum number of outstanding paged search requests allowed simultaneously.
ibm-slapdReplRestrictedAccess		Yes	Used to control access to the replication topology entry. If it is set to true, then only the root admin, local admin group members, and the master DN have access to the replication topology entry. Otherwise, any user with proper ACL might have access to the replication topology entry.
ibm-slapdReplVersion		Yes	This attribute defines the current version of the advanced replication feature.
ibm-slapdSortKeyLimit		Yes	Maximum number of sort keys that can be specified on a single sorted search request.
ibm-slapdSortSrchAllowNonAdmin		Yes	Whether the server should allow non-Administrators to request sorted search results.
ibmAttributeTypes		No	IBM attribute types.
ldapSyntaxes		No	LDAP syntaxes.
matchingRules		No	LDAP matching rules.

Table 103. Search operational attributes (continued)

Attribute name	User modifiable	Filter predicate	Description
matchingRuleUse		No	LDAP matching rule uses.
modifiersName	NO-USER-MODIFICATION	Yes	Name of last entry modifier.
modifyTimestamp	NO-USER-MODIFICATION	Yes	Time of last entry modification.
nameForms		No	Directory name forms.
objectClasses		No	LDAP object classes.
ownerPropagate		Yes[1]	Defines entry owner subtree propagation.
ownerSource	NO-USER-MODIFICATION	No	Source of the owner for an entry.
pwdAccountLockedTime		Yes	Specifies the time that the users account was locked.
pwdChangedTime		Yes [2]	Specifies the last time the entry password was changed.
pwdExpirationWarned		Yes	The time the user was first warned about the pending expiration of the password.
pwdFailureTime		Yes	The time stamps of the last consecutive authentication failures.
pwdGraceUseTime		Yes	The time stamps of the grace log in when the password expires.
pwdHistory		Yes	The history of users passwords.
pwdReset		Yes	Indicates that the password is reset.
replicaBindDN		Yes	Specifies the replica bind DN.
replicaBindMethod		Yes	Specifies the replica bind method.
replicaCredentials **replicaBindCredentials**		Yes	Specifies the replica bind credentials.
replicaHost		Yes	Specifies the replica host name.
replicaPort		Yes	Specifies the replica bind port.
replicaUpdateTimeInterval		Yes	Specifies replication update interval.
replicaUseSSL		Yes	Specifies SSL usage when binding to replica.
subschemaSubentry	NO-USER-MODIFICATION	No	Schema that is associated with an entry.
subtreeSpecification		No	Subtree specification.
Root DSE Attributes			
altServer		No	Alternate LDAP server.
changeLog	NO-USER-MODIFICATION	No	Distinguished name of the server change log.
firstChangeNumber	NO-USER-MODIFICATION	No	Change number for the earliest entry in the server change log.
ibm-enabledCapabilities	NO-USER-MODIFICATION	No	Capabilities that are enabled for use on this server.

Table 103. Search operational attributes (continued)

Attribute name	User modifiable	Filter predicate	Description
ibm-saslDigestRealmName	NO-USER-MODIFICATION	No	DIGEST-MD5 realm names for this server.
ibm-serverId	NO-USER-MODIFICATION	No	Advertises in the Root DSE the **ibm-slapdServerId** configuration option setting.
ibm-supportedCapabilities	NO-USER-MODIFICATION	No	Capabilities that are supported by this server.
ibm-supportedReplicationModels	NO-USER-MODIFICATION	No	Advertises in the root DSE the OIDs of replication models that are supported by the server.
ibmDirectoryVersion	NO-USER-MODIFICATION	No	Version of this directory server.
lastChangeNumber	NO-USER-MODIFICATION	No	Change number for the latest entry in the server change log.
ldapServiceName	NO-USER-MODIFICATION	No	LDAP service name for this server as `host@realm`.
namingContexts		No	LDAP server naming contexts.
supportedControl		No	Controls supported by this server.
supportedExtension		No	Extensions that are supported by this server.
supportedLDAPVersion		No	LDAP protocol versions that are supported by this server.
supportedSASLMechanisms		No	SASL mechanisms that are supported by this server.
vendorName	NO-USER-MODIFICATION	No	Name of the company that implemented the LDAP server.
vendorVersion	NO-USER-MODIFICATION	No	Version of the LDAP Server implementation.

Foot notes:
[1]ACL related attributes: This attribute is sometimes present in the entries, and sometimes it is derived. Use of this attribute in a filter predicate only matches when the attribute is stored in the entry.
[2]Password policy related attributes: This attribute is sometimes present in the entries, and sometimes it is derived. Use of this attribute in a filter predicate only matches when the attribute is stored in the entry.

Appendix I. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS V2R2 ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that the screen reader is set to read out

punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element
The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element
The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the

default option for the FILE keyword. In the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

*** indicates an optional syntax element that is repeatable**

The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3* , 3 HOST, 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write HOST STATE, but you cannot write HOST HOST.
3. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

+ indicates a syntax element that must be included

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

This information was developed for products and services offered in the U.S.A. or elsewhere.

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 character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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:

Site Counsel
IBM Corporation
2455 South Road
Poughkeepsie, NY 12601-5400
USA

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.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

COPYRIGHT LICENSE:

This information might contain sample application programs in source language, which illustrate 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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS™, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted

for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (http://www.ibm.com/software/support/systemsz/lifecycle/)
- For information about currently-supported IBM hardware, contact your IBM representative.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available at Copyright and Trademark information (http://www.ibm.com/legal/copytrade.shtml).

Glossary

This glossary includes terms and definitions for z/OS LDAP documentation. If you do not find the term that you are looking for, or to view glossaries for other IBM products, go to http://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, which are 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 can be purchased from *www.ansi.org*.
- *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 that are published by the International Telecommunication Union, Geneva, 1978.
- Open Software Foundation (OSF).

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 See *access control list (ACL)*.

ACL Filter
An **aclEntry** or **entryOwner** value that includes permissions for the **aclFilter** or **ownerFilter** scopes.

API Application program interface.

application program interface (API)
A functional interface supplied by the operating system or by a separately orderable licensed program that allows an application program written in a high-level language to use specific data or functions of the operating system or the licensed program.

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 attributes type and a sequence of one or more attribute values, each accompanied by an integer denoting the values syntax.

backend
A subsystem of the LDAP server which implements access to a persistent storage mechanism for information.

base ACL
Any **aclEntry** or **entryOwner** value that includes permissions for the **access-id, group**, or **role** scopes.

base effective permission
The access granted to a bound user before the application of any matching filter permissions. Before z/OS V1R12, this was the effective ACL.

bind identity
The bind identity for a bound user consists of their associated bind DN and alternate DNs.

After a successful bind request, a bind DN is associated with the bound user. Depending on the bind method (simple, CRAM-MD5, DIGEST-MD5, Kerberos, or certificate bind), there can also be a list of alternate DNs associated with the bound user.

bind information
Before z/OS V1R12, the bind information for a user included the bind identity and associated groups. Starting with z/OS V1R12, the bind information also includes

the client connection IP address, bind mechanism, and whether encryption was used.

binding
A relationship between a client and a server involved in a remote procedure call.

CCA Common Cryptographic Architecture.

CDBM
A file-based backend that stores configuration information.

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 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.

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 US 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 users 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 entry access information
Includes the access time of day and access day of week.

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.

DNS Domain Name System.

Domain Name System (DNS)
In the Internet suite of protocols, the distributed database system used to map domain names to IP addresses.

effective permission
The access granted to a bound user. In z/OS V1R12, the effective permission becomes finalized after any matching filter permissions are applied.

environment variable
A variable included in the current software environment that is available to any called program that requests it.

extended operations
A generic operation that extends the LDAP protocol. The operation contains an object identifier that uniquely identifies the intended operation. The extended operation allows additional operations to be defined for services not available elsewhere in the LDAP V3 protocol.

EXOP A backend that supports various LDAP extended operations. No entries are kept in this backend.

GDBM
A backend that manages change log entries created because of changes to the LDAP schema or to entries in other backends (including RACF entries). The change log entries can be kept in z/OS UNIX System Services files or in a DB2 database.

Generic Security Service (GSS) API
An application programming interface enabling application programs that do not implement remote procedure calls (RPCs) to have security services provided by a server in a computing environment. The GSSAPI provides security services to callers through a generic method that functions independently of underlying cryptography mechanisms or communication protocols and can thus be used in many different environments. IBM TDS for z/OS uses GSSAPI to perform Kerberos authentications.

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

JCL Job control language.

Job control language (JCL)
A control language used to identify a job to an operating system and to describe the jobs requirements.

Kerberos
The security system of the Massachusetts Institute of Technology's (MIT's) Project Athena. It uses symmetric key cryptography to provide security services to users in a network.

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.

LDAP Lightweight Directory Access Protocol.

LDBM
A general-purpose backend that stores its entries in z/OS UNIX System Services files.

Lightweight Directory Access Protocol (LDAP)
A client/server protocol for accessing a directory service.

master replica
The first instance of a specific directory in the namespace. After copies of the directory are 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.

NOID Numeric object identifier

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 z/OS UNIX System Services environment.

z/OS Cryptographic Services
A z/OS offering that supplies a set of interfaces for cryptographic functions.

programming interface
The supported method through which customer programs request software services. The programming interface consists of a set of callable services provided with the product.

protocol
A set of semantic and syntactic rules that determines the behavior of functional units in achieving communication.

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 that 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.

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.

SASL Simple Authentication Security Layer.

schema
See *directory schema*.

SDBM
A backend that provides access to RACF data. All entries are kept in RACF.

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.

SSHA Salted Secure Hash Algorithm
An enhanced version of the Secure Hash

Algorithm (SHA), that uses a randomly generated salt or seed, to better hash the resulting data.

SHA-2 – Secure Hash Algorithm Version 2
Published by NIST in Pub180-2: *Secure Hash Standard: Federal Information Processing Standards Publication 180-2* and consists of SHA224, SHA256, SHA384, and SHA512 hashing algorithms.

SSHA-2 Salted Secure Hash Algorithm Version 2
An enhanced version of the SHA-2 hashing algorithms that uses a randomly generated salt or seed to randomize the resulting hash.

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.

TDBM
A general-purpose backend that stores its entries in a DB2 database.

TDS Tivoli Directory Server.

thread A single sequential flow of control within a process.

Tivoli Directory Server (TDS)
Implementations of LDAP servers, clients, and utilities provided by IBM, supporting many IBM and non-IBM platforms.

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 allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. TLS is based on SSL Version 3.0

TSO Time Sharing Option.

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.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.

X/OPEN Directory Service (XDS)
An application programming interface that DCE uses to access its directory service components. XDS provides facilities for adding, deleting, and looking up names and their attributes. The XDS library detects the format of the name to be looked up and directs the calls it receives to either GDS or CDS. XDS uses the X/OPEN object management (XOM) API to define and manage its information.

X/OPEN object management (XOM)
An interface for creating, deleting, and accessing objects containing information. It is an object-oriented architecture: each object belongs to a particular class, and classes can be derived from other classes inheriting the characteristics of the original classes. The representation of the object is not apparent to the programmer; the object can be manipulated only through the XOM interface.

XOM The X/OPEN Object Management API.

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

Product Number: 5650-ZOS

Printed in USA

SC23-6788-01