z/VM
7.4

TCP/IP LDAP Administration Guide

IBM

Note:

Before you use this information and the product it supports, read the information in "Notices" on page 387.

Contents

Figures

Tables

About this document

This document is about administering the Lightweight Directory Access Protocol (LDAP) server on IBM z/VM, which includes such tasks as:

- Setting up schemas
- Modifying DN operations
- Accessing security information and authenticating
- Using access control
- Replication directories
- Creating aliases
- Configuring change logs
- Setting up referrals
- Organizing the directory namespace.

Intended audience

This document is intended to assist Lightweight Directory Access Protocol (LDAP) administration. This document is also intended for anyone who implements the directory service.

To do LDAP administration, you should be experienced in and have previous knowledge of directory services. You should have a good understanding of the TCP/IP in general and how z/VM implements the TCP/IP protocol suite. Also, you should understand the Lightweight Directory Access Protocol (LDAP).

Conventions and terminology

This topic describes important terminology and style conventions used in this document.

How the term "internet" is used in this document

In this document, an internet is a logical collection of networks supported by routers, gateways, bridges, hosts, and various layers of protocols, which permit the network to function as a large, virtual network.

Note: The term "internet" is used as a generic term for a TCP/IP network, and should not be confused with the Internet, which consists of large national backbone networks (such as MILNET, NSFNet, and CREN) and a myriad of regional and local campus networks worldwide.

How to Read Syntax Diagrams

This section describes how to read the syntax diagrams in this document.
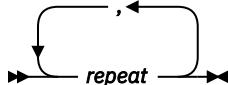
Getting Started: To read a syntax diagram, follow the path of the line. Read from left to right and top to bottom.
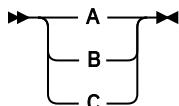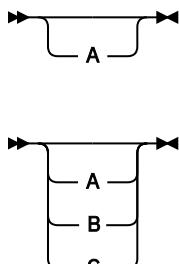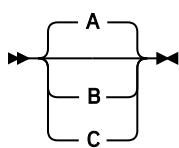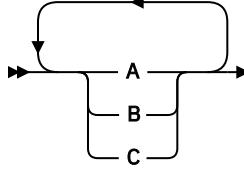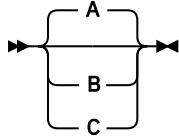
- The ►►—— symbol indicates the beginning of a syntax diagram.
- The ——► symbol, at the end of a line, indicates that the syntax diagram continues on the next line.
- The ►—— symbol, at the beginning of a line, indicates that a syntax diagram continues from the previous line.
- The ——►◄ symbol indicates the end of a syntax diagram.

Syntax items (for example, a keyword or variable) may be:

- Directly on the line (required)
- Above the line (default)

- Below the line (optional).

Syntax Diagram Description	Example
Abbreviations:	►► KEYWOrd ►◄
Uppercase letters denote the shortest acceptable abbreviation. If an item appears entirely in uppercase letters, it cannot be abbreviated.	
You can type the item in uppercase letters, lowercase letters, or any combination.	
In this example, you can enter KEYWO, KEYWOR, or KEYWORD in any combination of uppercase and lowercase letters.	

Symbols:

You must code these symbols exactly as they appear in the syntax diagram.

*	Asterisk
:	Colon
,	Comma
=	Equal Sign
-	Hyphen
()	Parentheses
.	Period

Variables:

Highlighted lowercase items (*like this*) denote variables.

In this example, *var_name* represents a variable you must specify when you code the KEYWORD command.

Example: ►► KEYWOrd ── *var_name* ►◄

Repetition:

An arrow returning to the left means that the item can be repeated.

A character within the arrow means you must separate repeated items with that character.

A footnote (1) by the arrow references a limit that tells how many times the item can be repeated.

Notes:

1 Specify *repeat* up to 5 times.

Syntax Diagram Description	Example
Required Choices: When two or more items are in a stack and one of them is on the line, you *must* specify one item. In this example, you must choose A, B, or C.	A B C
Optional Choice: When an item is below the line, the item is optional. In this example, you can choose A or nothing at all. When two or more items are in a stack below the line, all of them are optional. In this example, you can choose A, B, C, or nothing at all.	A A B C
Defaults: Defaults are above the line. The system uses the default unless you override it. You can override the default by coding an option from the stack below the line. In this example, A is the default. You can override A by choosing B or C.	A B C
Repeatable Choices: A stack of items followed by an arrow returning to the left means that you can select more than one item or, in some cases, repeat a single item. In this example, you can choose any combination of A, B, or C.	A B C
Syntax Fragments: Some diagrams, because of their length, must fragment the syntax. The fragment name appears between vertical bars in the diagram. The expanded fragment appears in the diagram after a heading with the same fragment name. In this example, the fragment is named "A Fragment."	▶▶─ A Fragment ─▶◀ **A Fragment** A B C

Where to Find More Information

Other z/VM manuals contain information about LDAP:

- For information about configuring the LDAP server, see *z/VM: TCP/IP Planning and Customization*.
- LDAP client utilities are documented in *z/VM: TCP/IP User's Guide*.
- Information about LDAP messages is in *z/VM: TCP/IP Messages and Codes*.

Appendix E, "Abbreviations and acronyms," on page 383, lists the abbreviations and acronyms that are used throughout this document.

For more information about related publications, see the documents listed in the "Bibliography" on page 391.

Links to other documents and websites

The PDF version of this document contains links to other documents and websites. A link from this document to another document works only when both documents are in the same directory or database, and a link to a website works only if you have access to the Internet. A document link is to a specific edition. If a new edition of a linked document has been published since the publication of this document, the linked document might not be the latest edition.

How to provide feedback to IBM

We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information. See How to send feedback to IBM for additional information.

Summary of changes for z/VM: TCP/IP LDAP Administration Guide

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line (|) to the left of the change.

SC24-6329-74, z/VM 7.4 (June 2025)

This edition includes changes to support product changes provided or announced after the general availability of z/VM 7.3 and 7.4.

[7.4 PH65378, 7.3 PH65377] z/VM TCP/IP support for EQDIO

With the PTFs for APARs PH65378 (7.4) and PH65377 (7.3), z/VM 7.4 and 7.3 provide a native network device driver for the z/VM TCP/IP stack that uses EQDIO adapters for network transport.

For more information, see:

- *z/VM: TCP/IP Planning and Customization*
- *z/VM: TCP/IP User's Guide*
- *z/VM: TCP/IP Messages and Codes*
- *z/VM: Installation Guide*
- *z/VM: Performance*

SC24-6329-74, z/VM 7.4 (September 2024)

This edition supports the general availability of z/VM 7.4. Note that the publication number suffix (-74) indicates the z/VM release to which this edition applies.

Removal of support for LAN Channel Station (LCS) emulation

Support for the OSE CHPID type, which is used to provide LAN Channel Station (LCS) emulation, is discontinued. TCP/IP no longer supports the LCS device driver. LCS documentation is removed from z/VM publications.

This satisfies the Statement of Direction from the z/VM 7.3 product announcement.

The following topic is updated:

- Appendix E, "Abbreviations and acronyms," on page 383

SC24-6329-73, z/VM 7.3 (December 2023)

This edition includes terminology, maintenance, and editorial changes.

SC24-6329-73, z/VM 7.3 (September 2023)

This edition supports product changes that were provided or announced after the general availability of z/VM 7.3.

[PH56199, VM66698] System SSL z/OS 2.5 equivalence

With the PTFs for APARs PH56199 (TCP/IP) and VM66698 (LE), z/VM 7.3 provides an update to the cryptographic services library, which includes certificate diagnostic enhancements and improved algorithmic support and allows for enablement of TLS 1.3, for secure connectivity to the z/VM platform.

SC24-6329-73, z/VM 7.3 (September 2022)

This edition supports the general availability of z/VM 7.3. Note that the publication number suffix (-73) indicates the z/VM release to which this edition applies.

Chapter 1. Introducing the LDAP server

The z/VM Lightweight Directory Access Protocol (LDAP) server 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:

- LDAP 2 and 3 support
- Interoperability with any LDAP 2 or 3 directory client
- Secure Sockets Layer (SSL) communication (SSL 3 and TLS 1)
- Client and server authentication using SSL/TLS
- Start TLS (Transport Layer Security) activation of secure communication
- Access controls on directory information using dynamic, nested, and static groups
- LDAP access to information stored in RACF®
- Challenge-Response Authentication Method (CRAM-MD5) and DIGEST-MD5 authentication
- Native authentication
- Aliases
- Basic and advanced replication
- Change logging
- Password encryption or hashing
- Password policy
- Root DSE information
- Referrals
- Schema publication and update
- Plug-in support to extend the 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 usually implement the complicated transaction or rollback schemes that relational databases use for doing high-volume complex updates. Directory updates are typically simple all-or-nothing changes, if they are allowed at all. Directories are tuned to give quick-response to high-volume lookup or search operations. They may have the ability to replicate information widely in order to increase availability and reliability, while reducing response time. When directory information is replicated, temporary inconsistencies between the replicas are considered acceptable, as long as they get in sync eventually.

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

What is LDAP?

The LDAP server's model for the directory service is based on a global directory model called Lightweight Directory Access Protocol (LDAP). LDAP 2 and LDAP 3, both supported in z/VM, are directory service protocols that run over TCP/IP. The details of LDAP 2 are defined in Internet Engineering Task Force (IETF) Request for Comments (RFC) 1777, *The Lightweight Directory Access Protocol*, and the details of LDAP 3 are defined in IETF RFCs 2251 through 2256. For a list of supported RFCs, see Appendix D, "Related Protocol Specifications," on page 377.

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 e-mail address. The values depend on what type of attribute it is. For example, a **mail** attribute might contain an e-mail address with an attribute value of `thj@vnet.ibm.com`. A `jpegPhoto` attribute would contain a photograph in binary JPEG format.

How is the information arranged?

In LDAP, directory entries are arranged in a hierarchical tree-like structure that sometimes reflects political, geographic or organizational boundaries. Entries representing countries appear at the top of the tree. Below them are entries representing states or national organizations. Below them might be entries representing people, organizational units, printers, documents, or just about anything else you can think of. Figure 1 on page 3 shows an example LDAP directory tree.

LDAP Directory Content

"root"

c=US
objectclass=country
c=US

c=UK

o=IBM
objectclass=organization
o=IBM

o=Lotus

o=Tivoli

cn=Tim Jones
objectclass=person
cn=Tim Jones
mail=thj@vnet.ibm.com
mail=jonesth@us.ibm.com

RDN: cn=Tim Jones
DN: cn=Tim Jones, o=IBM, c=US

- All entries have attributes (and values)
- objectclass is an attribute in all entries
- Attributes grouped into required and allowed
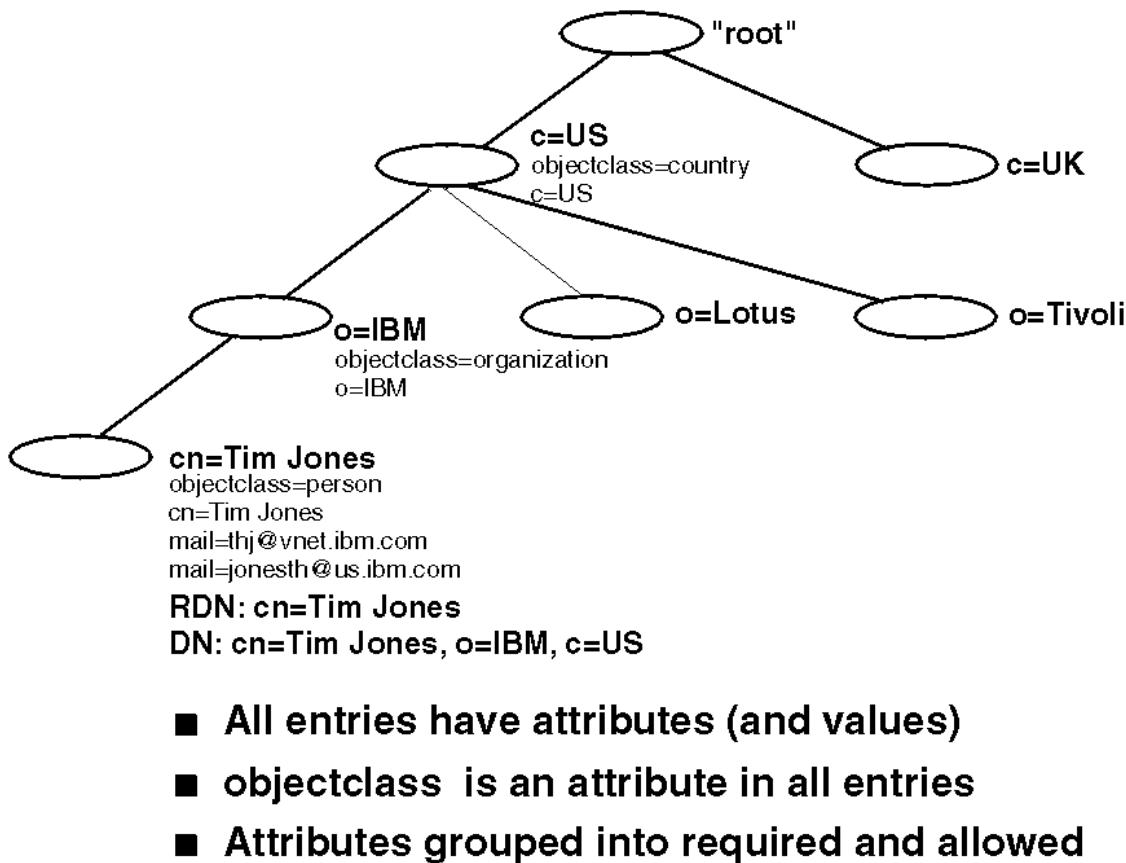
Figure 1. Directory hierarchy example

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=HAL, c=US`. The full DN format is described in IETF RFC 2253, *LDAP (V3): UTF-8 String Representation of Distinguished Names*.

The 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=hal,dc=com`. These naming formats can be mixed as well, for example: `cn=Tim Brown,ou=Sales,dc=hal,dc=com`.

How is the information accessed?

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

An example of search is, you might want to search the entire directory subtree below IBM for people with the name Tim Jones, retrieving the e-mail address of each entry found. LDAP lets you do this easily. Or you might want to search the entries directly below the c=US entry for organizations with the string Acme in their name, and that have a FAX number. LDAP lets you do this too. The section "How does LDAP work?" on page 4 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 need to 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/VM LDAP server?

You can use the z/VM LDAP server to provide a directory service of your very own. Your directory can contain just about anything you want to put in it. Some of the z/VM 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/VM LDAP server can respond

to requests for many logically different portions of the LDAP tree. A z/VM LDAP server can be configured to provide access to RACF, as well as store application-specific information.

- **Robust general-purpose databases**. The LDAP server comes with an LDBM backend. There are no restrictions on the types of information that this backend can contain. 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 Byte File System (BFS) files.

- **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. The LDAP server's access to RACF is managed by an additional configurable backend called SDBM. For more information, see Chapter 5, "Accessing RACF information," on page 61.

  Note: To use SDBM for ONLY authentication (LDAP bind processing), any security manager implementing the SAF service required by the `__passwd()` function call can be used. To use SDBM for accessing and updating user, 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. For more information, see CDBM backend configuration and policy entries in *z/VM: TCP/IP Planning and Customization*.

- **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. For more information on access control, see Chapter 10, "Using access control," on page 145. For more information about groups, see Chapter 9, "Static, dynamic, and nested groups," on page 133.

- **Threads**. The LDAP server is threaded for optimal performance. A single multi-threaded z/VM LDAP server process handles all incoming requests, reducing the amount of system overhead required.

- **Basic replication**. The LDAP server can be configured to maintain replica copies of its database. Master/consumer replication scheme is vital in high-volume environments where a single LDAP server just does not provide the necessary availability or reliability. Peer-to-peer replication is also supported. For more information, see Chapter 11, "Basic replication," on page 177. 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 in an LDBM or CDBM backend to be replicated to other servers. For more information, see Chapter 12, "Advanced replication," on page 193.

- **Referrals**. The LDAP server provides the ability to refer clients to additional directory servers. Using referrals you can distribute processing overhead, distribute administration of data along organizational boundaries, and provide potential for widespread interconnection beyond an organization's own boundaries. For more information, see Chapter 15, "Referrals," on page 273.

- **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. For more information, see Chapter 13, "Alias," on page 257.

- **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 an LDBM or CDBM backend, to the LDAP server schema, or to a RACF user, group, connection, or resource profile. For more information, see Chapter 14, "Change logging," on page 265.

- **Configuration**. The LDAP server is highly configurable through a single configuration file which allows you to change just about everything you would ever want to change. Configuration options have reasonable defaults, making your job much easier. For more information, see Configuring the LDAP Server in *z/VM: TCP/IP Planning and Customization*.

- **Secure communication**. The LDAP server can be configured to encrypt data to and from LDAP clients using SSL. The LDAP server supports the Start TLS extended operation to switch a non-secure connection to a secure connection. It has a variety of ciphers for encryption to choose from, all of which provide server and optionally client authentication through the use of X.509 certificates. For more information, see Setting up for SSL/TLS in *z/VM: TCP/IP Planning and Customization*.
- **Native authentication**. The z/VM LDAP server allows clients to bind to entries in an 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 an LDBM or CDBM backend along with a security manager-maintained password. Password authentication is then performed by the security manager. For more information, see Native authentication in *z/VM: TCP/IP Planning and Customization*.

  Note: To use native authentication, any security manager implementing the SAF service required by the **__passwd()** function call can be used.
- **LDAP 3 support**. The LDAP server provides support for LDAP 2 and 3. LDAP 3 includes:
  - All protocol operations
  - Implicit bind
  - Certificate (or Simple Authentication and Security Layer) bind
  - LDAP 3 referrals
  - Aliases
  - Controls
  - Root DSE support
  - Internationalization (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. For more information, see Chapter 3, "LDAP directory schema," on page 13.
- **Internationalization (UTF-8) support**. The LDAP server allows storage, update and retrieval, through LDAP operations, of national language data using LDAP 3. For more information, see Internationalization Support in *z/VM: TCP/IP Planning and Customization*.
- **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 communication. In order to use a certificate on bind, the server must be configured to perform both client and server authentication. This configuration ensures both entities are who they claim to be. For more information, see Setting up for SSL/TLS in *z/VM: TCP/IP Planning and Customization*.
- **SASL CRAM-MD5 and DIGEST-MD5 authentication**. The LDAP server allows clients to bind to the server using DIGEST-MD5 (RFC 2831) and CRAM-MD5 (Challenge-Response Authentication Method - RFC 2195) authentication bind methods. For more information, see Chapter 8, "CRAM-MD5 and DIGEST-MD5 authentication," on page 129.
- **Support for root DSE**. The LDAP server supports search operations, including subtree search, against the root of the directory tree as described in IETF RFC 2251, *The Lightweight Directory Access Protocol (V3)*. The so-called Root DSE can be accessed using LDAP V3 search operations. For more information, see "Root DSE" on page 293.
- **Extended group membership searching**. The LDAP server supports extended group membership searching which allows the LDAP server to find a DN that may be a member of static and nested groups in a backend (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. For more information about the `extendedGroupSearching` configuration file option, see extendedGroupSearching in *z/VM: TCP/IP Planning and Customization*.
- **Supported server controls**. The LDAP server supports the following:

authenticateOnly
Do Not Replicate
IBMModifyDNRealignDNAttributesControl
IBMModifyDNTimelimitControl
IBMSchemaReplaceByValueControl
manageDsaIT
No Replication Conflict Resolution
pagedResults
PasswordPolicy
PersistentSearch
Refresh Entry
replicateOperationalAttributes
Replication Supplier ID Bind
Server Administration
SortKeyRequest

For more information, see Appendix B, "Supported server controls," on page 345.

- **Supported extended operations**.

  Account status
  Cascading control replication
  changeLogAddEntry
  Control replication
  Control replication error log
  Control replication queue
  Effective password policy
  Quiesce or unquiesce context
  Replication topology
  Start TLC
  unloadRequest
  User type

For more information about extended operations, see Appendix C, "Supported extended operations," on page 357.

- **Attribute encryption**. The LDAP server supports encryption of the values of several critical attributes to prevent unauthorized access to these attribute values in LDBM and CDBM backends. The attributes that can be encrypted are as follows:

  ibm-replicaKeyPwd
  ibm-slapdAdminPw
  ibm-slapdMasterPw
  replicaCredentials
  secretKey
  userPassword

For more information, see Configuring for Encryption or Hashing in *z/VM: TCP/IP Planning and Customization*.

- **Multiple socket ports**. The LDAP server can be configured to listen for secure and nonsecure connections from clients on one or more IPv4 or IPv6 interfaces on a system. With the **listen** configuration option on the LDAP server, the hostname, 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. For more information, see listen in *z/VM: TCP/IP Planning and Customization*.

- **Persistent search**. The LDAP server provides an event notification mechanism for applications, directories, and meta directories that need to maintain a cache of directory information or to

synchronize directories when changes are made to an LDAP directory. Persistent search will allow these applications to be notified when a change has occurred. For more information, see Appendix B, "Supported server controls," on page 345.

- **ibm-entryuuid** attribute. The LDAP server now generates a unique identifier for any entry that is created or modified and does not already have a unique identifier assigned. The unique identifier is stored in the **ibm-entryuuid** attribute. The **ibm-entryuuid** attribute is replicated to servers that support the **ibm-entryuuid** attribute. To configure the **serverEtherAddr** option in the LDAP server configuration file, see serverEtherAddr in *z/VM: TCP/IP Planning and Customization*.

- **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**, and **ibm-entryuuid**.

- **ibm-allMembers** and **ibm-allGroups**. The LDAP server now supports the querying of the members of static, dynamic, and nested groups in an 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 attributes.

- **Plug-in support**: The LDAP server can be configured with extensions called plug-ins. The plug-ins are supplied by other products or created by you. Plug-ins are invoked before, during, or after the LDAP server processes a client request. For more information on configuring a plug-in, see Configuring the LDAP Server in *z/VM: TCP/IP Planning and Customization*. For information about creating a plug-in, see *z/VM: TCP/IP Programmer's Reference*.

- **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**. For more information, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

- **Password policy**. The LDAP server supports password policy, which is a set of rules that control how passwords are defined, used, and administered. For more information, see Chapter 7, "Password policy," on page 103.

- **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 136 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 347 and "SortKeyRequest" on page 354 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 6, "Administrative group and roles," on page 91 for more information.

Chapter 2. Data model

The LDAP data model is closely aligned with the X.500 data model. In this model, a directory service provides a hierarchically organized set of *entries*. Each of these entries is represented by an *object class*. The object class of the entry determines the set of *attributes* which are required to be present in the entry as well as the set of attributes that can optionally appear in the entry. An attribute is represented by an *attribute type* and one or more *attribute values*. In addition to the attribute type and values, each attribute has an associated *syntax* which describes the format of the attribute values. Examples of attribute syntaxes for LDAP 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 making up the RDN for an entry must also be present as an attribute=value pair or pairs in the entry. This is not true of the other components of the DN. The 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 something like:

```
cn=John Smith+studentNumber=123456.
```

The plus sign (+) is used to delimit separate attribute=value pairs within an RDN. The entry's DN might look like:

```
cn=John Smith+studentNumber=123456, ou=downtown, o=Big University, c=US
```

Any attribute can be used to make up an RDN except:

- attributes with binary syntax, UTC time syntax, or generalized time syntax.

    Note: The **userPassword** attribute is binary, therefore, it cannot appear in an RDN. Time stamp attributes use one of the time syntaxes, therefore, they 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 IETF RFC 2253 *LDAP (v3): UTF-8 String Representation of Distinguished Names*. A semicolon (;) character may be used to separate RDNs in a distinguished name, although the comma (,) character is the typical notation. 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 pound sign (#) character occurring at the beginning of the string
- A space character occurring at the end of the string
- One of the characters
  - apostrophe (')
  - equal sign (=)
  - plus sign (+)
  - backslash (\)
  - less than sign (<)
  - greater than sign (>)
  - semicolon (;)

Alternatively, a single character to be escaped may be prefixed by a backslash (\). This method may be used to escape any of the characters listed above, plus the quotation mark. Pound 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 designed to be convenient for common forms of name. This section gives a few examples of distinguished names written using this notation. First is a name containing three components:

```
OU=Sales+CN=J. Smith,O=Widget Inc.,C=US
```

This example shows a method of escaping a comma in an organization name:

```
CN=R. Smith,O=Big Company\, Inc.,C=US
```

Domain component naming

Domain component naming as specified by RFC 2247 is also supported in the LDAP server. For example, the domain name `ibm.com` could be specified as an entry in the LDAP server with the following 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 3. LDAP directory schema

LDAP 3, as defined in IETF RFC 2252 *Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions* and IETF RFC 2256 *A Summary of the X.500(96) User Schema for use with LDAPv3*, describes schema publication and update. Schema publication provides the ability to query the active directory schema through the use of the LDAP search function. Schema update is the ability to change the schema while the directory server is running.

Note:

- The z/VM 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 may 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 will save the schema entry. The default is **/var/ldap/schema**. This directory should be backed up as part of the normal system backup procedure since the loss of the schema directory will invalidate 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.

- When the z/VM LDAP server is first started, the server supplies an initial schema. This initial schema is sufficient for usage of the SDBM, CDBM (with configuration related entries), and GDBM backends, but needs to be updated for usage of LDBM and CDBM with user defined entries. The initial schema elements cannot be deleted and can be modified only in limited ways. For the contents of the initial schema, see Appendix A, "Initial LDAP server schema," on page 323.

- 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 the LDAP root or schema administrator, are subject to ACL checking. In particular for a basic 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 the LDAP root or schema administrator can update the schema. This ACL can be modified. See Chapter 10, "Using access control," on page 145 for more information.

Setting up the schema for LDBM 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 or CDBM. These files are USRSCHEM LDIF and IBMSCHEM LDIF and are located on the TCPMAINT 591 disk.

Determine which of these schema files, if any, could be used to represent the data to be stored in the LDBM or CDBM database, or locate or create other schema files to use. Note that the IBMSCHEM LDIF schema definitions require that the definitions contained in USRSCHEM LDIF are loaded prior to loading IBMSCHEM LDIF.

Use the LDAPMDFY command to load the schema. For example, the commands to load the USRSCHEM LDIF and IBMSCHEM LDIF schema files would be similar to:

```
ldapmdfy -h ldaphost -p ldapport -D adminDN -w passwd -f //usrschem.ldif
```

```
ldapmdfy -h ldaphost -p ldapport -D adminDN -w passwd -f //ibmschem.ldif
```

For more information about LDAPMDFY, see *z/VM: TCP/IP User's Guide*.

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 LDAPMDFY utility. For more information about the schemaReplaceByValue configuration option, see Configuring the LDAP Server in *z/VM: TCP/IP Planning and Customization*. For more information about the IBMSchemaReplaceByValueControl control, see "IBMSchemaReplaceByValueControl" on page 346.

2. When the LDAP schema is modified using the USRSCHEM LDIF and IBMSCHEM 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 needs to be made again. This includes any changes that are allowed to attributes and object classes in the initial LDAP schema.

Schema introduction

Entries in the directory are made up of attributes which consist of an attribute type and one or more attribute values. These are referred to as *attribute=value* pairs. Every entry contains one or more `objectclass=value` pairs that identify what type of information the entry contains. The object classes associated with the entry determine the set of attributes which must or may be present in the entry.

The z/VM 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 2. 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/VM 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 may be used interchangeably when an attribute type or object class definition specifies an object identifier. Most schema definitions use the textual name as the object identifier for these definitions.

Note: Non-numeric object identifiers, for example `myattr-oid`, can be used instead of numeric object identifiers.

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/VM LDAP server. These are listed in Table 4 on page 24, Table 5 on page 25, and Table 6 on page 26. 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 will fail. The syntaxes supported by the z/VM LDAP server are shown in Table 4 on page 24 and Table 5 on page 25.

    Matching rules may 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/VM LDAP server will perform evaluations to the best of its ability, but the results may not be as expected. The z/VM 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 1. Syntax and default EQUALITY, ORDERING, and SUBSTR matching rules

Syntax	EQUALITY	ORDERING	SUBSTR
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

Table 1. Syntax and default EQUALITY, ORDERING, and SUBSTR matching rules (continued)			
Syntax	**EQUALITY**	**ORDERING**	**SUBSTR**
Facsimile Telephone Number	telephoneNumberMatch	-	telephoneNumberSubstringsMatch
Fax	-	-	-
Generalized Time	generalizedTimeMatch	generalizedTimeOrderingMatch	-
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/VM LDAP server also verifies that the matching rules specified for EQUALITY, ORDERING, and SUBSTR are consistent with the specified SYNTAX. Table 2 on page 16 shows acceptable values EQUALITY, ORDERING, and SUBSTR.

Table 2. Syntax and acceptable matching rules (EQUALITY, ORDERING, and SUBSTR)			
Syntax	**EQUALITY**	**ORDERING**	**SUBSTR**
Attribute Type Description	objectIdentifierFirstComponentMatch	-	-
Binary	-	-	-
Bit String	bitStringMatch	-	-

Table 2. Syntax and acceptable matching rules (EQUALITY, ORDERING, and SUBSTR) (continued)			
Syntax	**EQUALITY**	**ORDERING**	**SUBSTR**
Boolean	booleanMatch caseIgnoreMatch caseExactMatch	-	-
Certificate	certificateMatch certificateExactMatch	-	-
Certificate List	-	-	-
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	-	-

Syntax	EQUALITY	ORDERING	SUBSTR
Table 2. Syntax and acceptable matching rules (EQUALITY, ORDERING, and SUBSTR) (continued)			
Matching Rule Description	objectIdentifierFirstComponentMatch	-	-
Matching Rule Use Description	objectIdentifierFirstComponentMatch	-	-
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	-	-	-

Syntax	EQUALITY	ORDERING	SUBSTR
Table 2. Syntax and acceptable matching rules (EQUALITY, ORDERING, and SUBSTR) (continued)			
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 may be inherited by specifying a superior attribute type. This is done by specifying the keyword SUP, followed by the object identifier of the superior attribute type. This is known as an attribute type hierarchy and referred to as inheritance. A superior hierarchy may be created with multiple levels of inheritance. In the following partial example, `ePersonName` and `personName` would inherit their SYNTAX from name.

```
ePersonName SUP personName
personName SUP name
name SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
```

When the SYNTAX, EQUALITY, ORDERING, or SUBSTR values are not specified for an attribute type, the attribute type hierarchy are used to determine these values. The SYNTAX must be specified on the attribute type or through inheritance.

The number of values that may be stored in each entry for an attribute type is limited to one value if the keyword SINGLE-VALUE is specified. Otherwise, any number of attribute values may exist in the entry.

The OBSOLETE keyword indicates that the attribute type cannot be used to add data to existing entries or to store data in new entries. Modifications to entries which contain data values of an attribute type which has been made obsolete will fail unless all data values for all obsolete attribute types are removed during the modification. Searches specifying the obsolete attribute type will return the entries containing the attribute type. If an obsolete attribute type is referred to in a superior hierarchy, the inherited values will continue to be resolved.

Example 1:

```
attributeTypes: ( 1.2.3.4 NAME 'obsattr1' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
OBSOLETE )
attributeTypes: ( 5.6.7.8 NAME 'validattr1' SUP obsattr1 )
```

would be the same as

```
attributeTypes: ( 5.6.7.8 NAME 'validattr' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

Example 2:

```
attributeTypes: ( 10.20.30.40 NAME 'obsattr2' SUP obsattr3 )
attributeTypes: ( 50.60.70.80 NAME 'obsattr3'
  EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
attributeTypes: ( 90.100.110.120 NAME 'validattr2' SUP obsattr2 )
```

would be the same as

```
attributeTypes: ( 90.100.110.120 NAME 'validattr2'
   EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

The USAGE keyword's valid values are `userApplications` or one of three operational values (`directoryOperation`, `distributedOperation`, or `dSAOperation`). An attribute type which has an operational USAGE value is called an operational attribute. Operational attributes are treated differently than non-operational attributes. In particular, the value of an operational attribute type in an entry is only returned by a search operation if the attribute type is specified in the list of attributes to be returned. 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`.

The z/VM 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 Chapter 11, "Basic replication," on page 177 for more information.

Note: LDAP 3 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/VM LDAP server, the additional information defined using this attribute is the ACCESS-CLASS of the associated attribute type.

  ACCESS-CLASS specifies the level of access users have to data values of this attribute type. The levels that may be specified for user-defined attribute types are `normal`, `sensitive`, and `critical`. The `system` and `restricted` keywords are for LDAP server use and are specified for some of the attribute types controlled by the server. See "Attribute access classes" on page 148 for the definition of access classes.

  Note: Other LDAP servers from IBM use the DBNAME and LENGTH characteristics to specify additional information for their implementations. These may be specified in the schema but are not used by the z/VM LDAP server.

- **Object classes**

  Object classes define the characteristics of individual directory entries. The object classes listed in a directory entry determine the set of required and optional attributes for the entry. Each object class defined in a schema must contain a unique numeric object identifier and optionally contain a textual name, zero or more alias names, a description of the object class, and lists of required (MUST) or optional (MAY) attribute types.

  Required and optional attribute types for an object class may be inherited by specifying one or more superior object classes in an object class definition. This is done by specifying the keyword SUP followed by the object identifiers of the superior object classes. This is known as an object class hierarchy and referred to as multiple inheritance. A superior hierarchy may be created with multiple levels of inheritance.

  Each object class is defined as one of three types: STRUCTURAL, ABSTRACT, or AUXILIARY. The type 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 can not be changed.** Once an entry is defined in the directory, it must be deleted and recreated to change the structural object class.

Abstract and auxiliary object classes are used to provide common characteristics to entries with different structural object classes. Abstract object classes are used to derive additional object classes. Abstract object classes must be referred to in a structural or auxiliary superior hierarchy. Auxiliary object classes are used to extend the set of required or optional attribute types of an entry.

When using the keyword **SUP** to create an object class hierarchy, an auxiliary class should only specify superior object classes that are either auxiliary or abstract object classes. Similarly, a structural object class should only specify superior object classes that are either structural or abstract object classes. If these rules are not followed, the z/VM 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 . The schema entry specifies `top`, `subEntry`, `subSchema`, and `ibmSubschema` as object classes. The object classes form the following hierarchy:
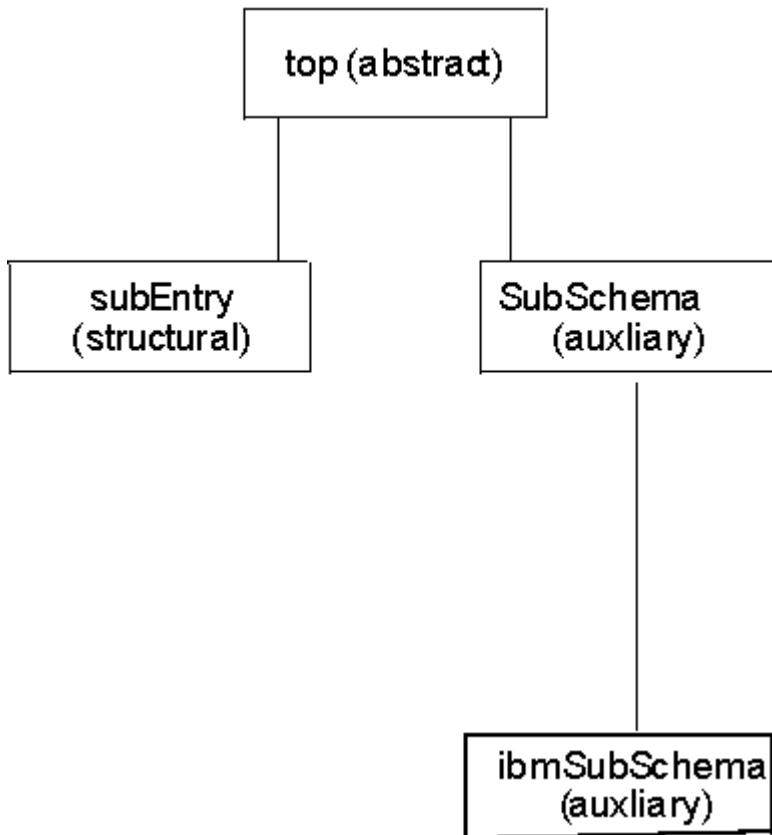
Figure 3. Object class hierarchy example

In this example, the `subEntry` object class is the base structural object class.

The `OBSOLETE` keyword indicates that the object class cannot be used to define entries in the directory. When an object class is made obsolete, new entries specifying the obsoleted object class cannot be added to the directory and existing entries cannot be modified unless the obsolete object class is removed from the entries' object class list. When the obsolete object class is removed from the entry, any attributes in the entry that are associated only with that object class must also be removed. These changes must be made through the same modify operation. If an obsolete object class is specified in a superior hierarchy for a new entry, then attempts to add the entry to the LDAP directory will fail.

- **LDAP syntaxes**

Each attribute type definition includes the LDAP syntax which applies to the values for the attribute. The LDAP syntax defines the set of characters which are allowed when entering data into the directory.

The z/VM LDAP server is shipped with predefined supported syntaxes. See Table 4 on page 24 and Table 5 on page 25 for the list of syntaxes supported by the z/VM LDAP server. The set of syntaxes cannot be changed, added to, or deleted by users.

- **Matching rules**

  Matching rules allow entries to be selected from the database based on the evaluation of the matching rule assertion. Matching rule assertions are propositions which may evaluate to true, false, or undefined concerning the presence of the attribute value or values in an entry.

  The z/VM LDAP server is shipped with predefined supported matching rules. See Table 6 on page 26 for the list of matching rules supported by the z/VM LDAP server. The set of matching rules cannot be changed, added to, obsoleted, or deleted by users.

Schema attribute syntax

The attributes which are used in the schema entry use specific character representations in their values. These character representations are described in Table 3 on page 22. The terms shown in this table are used in the schema attribute definitions in the next section.

Table 3. 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/VM LDAP server allows values to be any length, regardless of the specification of a length in the attribute type definition. User installations that want to limit the length of values need to handle this during data input.
numericoid	A dotted decimal string. Example: `2.5.13.72` **Note:** A non-numeric object identifier, for example `myattr-oid`, can be used instead of a numeric object identifier.
oid	A single object identifier. This may be specified either as a name or as a numeric object identifier. Examples: ```name 2.5.4.41```
oidlist	A list of object identifiers specified as names or numeric object identifiers separated by dollar signs ($) within parentheses. Example: ```(cn $ sn $ postaladdress $ 2.5.4.6)```

Term	Definition
Table 3. Character representations (continued)	
oids	An *oid* or *oidlist*.
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 LDAP 3 does not support an underscore character (_) as a valid character in a *descr*, the z/VM LDAP server allows the use of an underscore character to facilitate data migration. This use should be minimized whenever possible and may not be supported by other servers.
qdstring	A quoted descriptive string shown as `'dstring'`. The descriptive string (*dstring*) is composed of one or more UTF-8 characters.
	Example:
	``` 'This is an example of a quoted descriptive string.' ```

LDAP schema attributes

The five attributes used to define an LDAP schema are discussed below. For these schema attributes, the *numericoid* must be the first item in the definition. All other keywords and values may be in any order.

LDAP syntaxes

The set of syntaxes which are supported by the z/VM LDAP server cannot be modified, added to, or deleted by users. The descriptive material included here is for information aboutly.

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/VM LDAP server fall into two categories. The first set, as shown in , would be used when defining attribute types that are used for directory data.

Table 4. Supported LDAP syntaxes - general use		
Numeric object identifier	**Description**	**Valid values**
1.3.6.1.4.1.1466.115.121.1.5	Binary	Binary data
1.3.6.1.4.1.1466.115.121.1.6	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)
1.3.6.1.4.1.1466.115.121.1.24 **Note:** The effective time zone for the LDAP server is assumed when calculating GMT from local time.	Generalized Time	*yyyymmddhhmmss.ffffff* (local time) *yyyymmddhhmmss.ffffff*Z (GMT) *yyyymmddhhmmss.ffffff-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
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)

Table 4. Supported LDAP syntaxes - general use (continued)		
Numeric object identifier	**Description**	**Valid values**
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.

The second set of syntaxes defined by the z/VM 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 5. Supported LDAP syntaxes - server use	
Numeric object identifier	**Description**
1.3.6.1.4.1.1466.115.121.1.3	Attribute Type Description
1.3.6.1.4.1.1466.115.121.1.16	DIT Content Rule Description
1.3.6.1.4.1.1466.115.121.1.17	DIT Structure Rule Description
1.3.18.0.2.8.1	IBM Attribute Type Description
1.3.18.0.2.8.3	IBM Entry UUID Description
1.3.6.1.4.1.1466.115.121.1.54	LDAP Syntax Description
1.3.6.1.4.1.1466.115.121.1.30	Matching Rule Description
1.3.6.1.4.1.1466.115.121.1.31	Matching Rule Use Description
1.3.6.1.4.1.1466.115.121.1.35	Name Form Description
1.3.6.1.4.1.1466.115.121.1.37	Object Class Description
1.3.6.1.4.1.1466.115.121.1.58	Substring Assertion

Matching rules

The set of matching rules which are supported by the z/VM LDAP server cannot be modified, added to, obsoleted, or deleted by users. The descriptive material included here is for information aboutly.

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/VM LDAP server is a fixed set as listed in the following table.

Table 6. Supported matching rules		
Name	**Numeric object identifier**	**Assertion syntax**
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.
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.

Table 6. Supported matching rules (continued)

Name	Numeric object identifier	Assertion syntax
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 case-insensitive 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 6. Supported matching rules (continued)		
Name	**Numeric object identifier**	**Assertion syntax**
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 will result in a FALSE match status when used in a search filter.

5. The LDBM and CDBM backends ignore the **ORDERING** and **SUBSTR** matching rules and always use 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 may be inherited from the superior attribute type. The referenced superior attribute type must also be defined in the schema. When the SYNTAX, EQUALITY, ORDERING, or SUBSTR values are not specified for an attribute type, the attribute type hierarchy 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/VM LDAP server allows values to be any length, regardless of the specification of a length in the attribute type definition. User installations that want to manage the lengths of values need to handle this when values are put into the directory.

SINGLE-VALUE
    Limits entries to only one value for this attribute type.

NO-USER-MODIFICATION
    When specified, users may not modify values of this attribute type.

USAGE *attributeUsage*
    Specify `userApplications` for *attributeUsage*. If USAGE is not specified, the default is `userApplications`.

    The **directoryOperation**, **distributedOperation**, and **DSAOperation** keywords are used to create operational attributes. Operational attributes are treated differently than non-operational attributes. In particular, the value of an operational attribute type in an entry is only returned by a search operation if the attribute type is specified in the list of attributes to be returned. 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] )
```

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 148 for the definition of these values. The default attribute access class for an attribute is `normal`.

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 may also specify DBNAME, LENGTH, and other keywords and values. These keywords are not used by the z/VM LDAP server and do not need to be specified when creating schemas. If they are specified in a schema used by the z/VM 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 may be specified when adding or modifying a directory entry.

The `extensibleObject` object class is an AUXILIARY object class which allows an entry to optionally hold any attribute type. The `extensibleObject` object class is supported by the z/VM 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 and can be found at Internet Assigned Numbers Authority (IANA) (www.iana.org/).

Search the site for the "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 we have been assigned OID arc `1.3.18.0.2.1000.100`. (**Note**: Do not use this OID arc for defining your own schema elements. This arc is assigned to IBM for its use.) The following example adds a new object class that refers to two new attribute types. As you can see, the object class and attribute types can be added to the schema using a single LDAP modify operation. The changes to the schema are represented in LDIF mode input below:

```
dn: cn=schema
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/VM LDAP server is first started, the server supplies an initial schema. This initial schema is sufficient for usage of the SDBM, CDBM (with configuration related entries) and GDBM backends, but will need to be updated for usage of LDBM and CDBM with user defined entries. The schema files shipped with the LDAP server, USRSCHEM LDIF and IBMSCHEM LDIF, might be sufficient for your usage of LDBM or CDBM. (For more information about adding these files to the schema, see "Setting up the schema for LDBM and CDBM" on page 13.) If they are not sufficient, you can change the schema as needed. The schema entry is required and cannot be deleted. When deleting an attribute type or object class definition, you need to provide just the object identifier enclosed in parentheses. Any additional fields that are specified are checked for proper 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 very restricted. See "Changing the initial schema" on page 33 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 may 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 an LDBM or CDBM 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 323 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 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 would have to contain all the unchanged values from the schema in addition to the updated and new values so that no unchanged existing values are lost.

To address this problem, the LDAP server supports two different behaviors when using a modify with replace operation on the schema entry:

1. Standard RFC behavior, in which all the existing values for an attribute are replaced by the ones specified in the modify operation. 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 33.

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. For more information, refer to Configuring the LDAP Server in *z/VM: TCP/IP Planning and Customization*.

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. Refer to Appendix B, "Supported server controls," on page 345 for more information.

If neither the **schemaReplaceByValue** configuration option nor 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' ... )
```

We would like 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 isn't inadvertently removed from an existing attribute type.

Updating a numeric object identifier (NOID)

It may 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 that 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.

Example: suppose we want to change the NOID of the xyz attribute type from 1.3.5.7 to 2.4.6.8. The update file for the modify operation to accomplish this would look like:

```
cn=schema
-attributetypes=( 1.3.5.7 NAME 'xyz' DESC 'xyz attribute added for application abc' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 USAGE userApplications )
+attributetypes=( 2.4.6.8 NAME 'xyz' DESC 'xyz attribute added for application abc' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 USAGE userApplications )
```

Changing a NOID should not need to be done as part of normal LDAP server operations. It is intended to be used as an error recovery device for when an incorrect NOID has been added to the schema.

Analyzing schema errors

Following is some information about the possible cause of some schema errors that may be encountered when updating schema:

- For enhanced readability, *type:value* pairs in LDIF files may be split across multiple lines. The indicator to LDIF that the subsequent lines are continuations is that the first character on the subsequent line is a space. This character is ignored by parsers and it is assumed that the next character immediately follows the previous line. Therefore, if a space is needed between the last value on one line and the first value on the subsequent line, a second space needs to exist on the subsequent LDIF line. Various reason codes related to unrecognized values may be issued.

- Only limited changes are allowed to the initial schema, as described in "Changing the initial schema" on page 33. 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 will fail. For example,

```
 IBMAttributeTypes: ( 1.2.3.4 ACCESS-CLASS normal )
```

cannot be specified unless

```
 attributeTypes: ( 1.2.3.4 NAME 'sample' ... )
```

is also defined.

- While the UTC Time syntax is supported, usage of the Generalized Time syntax is recommended. For UTC Time syntax, year values between 70 and 99 assume 1970 to 1999 and values between 00 and 69 assume 2000 to 2069.

- When searching attribute type values of GMT or UTC Time syntax, use GMT syntax in the search filter rather than local time. All time values are stored in the data store as GMT times.

Retrieving the schema

The following sections describe how you can display the schema entry and also find the `subschemaSubentry` DN.

Displaying the schema entry

The following command shows how to search for the schema entry. Note that the scope must be **base** in the search request to display the schema.

```
ldapsrch -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 323. After the schema has been updated by the administrator, the search results will show the full schema as the union of the initial schema and the added schema elements.

The search results will 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.

```
ldapsrch -h ldaphost -p ldapport -s base -b "o=Acme Company, c=UK" "objectclass=*"
   subschemasubentry

o=Acme Company, c=UK
subschemasubentry=cn=schema
```

Chapter 4. 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 may result in unintentional or unwanted results.

In LDAP, modify DN operations are supported only in the LDBM (file-based) and CDBM (file-based) backends.

Modify DN operation syntax

The z/VM implementation of the Modify DN operation supports all required and optional parameters described for the operation in RFC 2251. Specifically, these parameters are required:

- **entryDN:** This is the Distinguished Name (DN) of the entry whose name will be changed. This entry may or may not have subordinate entries. This parameter may not be a zero-length string.
- **newRdn:** The Relative Distinguished Name (RDN) that will form the leftmost component of the new name of the entry. This parameter may not be a zero-length string. If the intent of the Modify DN operation is to move the target entry to a new superior without changing its RDN, the old RDN value must be supplied in the **newRdn** parameter. The attributes and values in the **newRdn** parameter are added to the entry if they are not already present in the entry.
- **deleteoldrdn:** A boolean parameter that controls whether the old RDN attribute values are to be retained attributes of the entry or whether they will be deleted from the entry.

The following parameter to the Modify DN operation is optional:

- **newSuperior:** The Distinguished Name (DN) of the entry which will become the immediate superior of the renamed entry (identified by the **entryDN** parameter). If this parameter is present, it may consist of a zero-length string or a non-zero-length string. See "Modify DN operations related to suffix DNs" on page 49 for more information on the use of a zero-length string for this parameter. A zero-length string value for this parameter ("") will signify that the new superior entry is the root DN.

This operation also supports optional values, or controls, to influence the behavior of the operation. Two controls are supported (see Appendix B, "Supported server controls," on page 345):

- **IBMModifyDNTimelimitControl:** This control causes the Modify DN operation to be abandoned if its duration exceeds the time limit represented by the control value expressed in seconds. No changes are made if the operation is abandoned. This control is honored even if it is set by the admin DN for the server. When this control is present, it will **not** be propagated to the replica servers. (See "Modify DN operations and replication" on page 57 for more information about replication of Modify DN operations.)
- **IBMModifyDNRealignDNAttributesControl:** This control causes the server to search for all attributes whose attribute type is based on a DN syntax (designated by OID 1.3.6.1.4.1.1466.115.121.1.12) and whose values match any of the old DN values being renamed as part of the Modify DN operation, and to modify the old DN values to reflect the corresponding renamed DN attribute values. This includes modifications to two other attribute types which have constructed DN-type attribute values (those whose attribute syntax is not distinguished name but which may be used to store DN values). They are **aclEntry** and ownership **entryOwner** attributes. Updates to constructed DN types will be limited to these two attributes defined by the LDAP Server. No changes will be made to any user constructed types.

  This control is an all-or-none operation in which the server attempts to realign all appropriately-matched DN attribute values in the LDBM or CDBM backend. Users cannot limit the scope of values which should be realigned. If a failure arises during the realignment operation, it realigns none of the

values, and the Modify DN operation fails. No changes are made if the operation is abandoned. It should be noted that even if the control is designated as non-critical, the server will still try to honor the intent of the control and if this attempt fails, the entire Modify DN operation will fail.

When **IBMModifyDNRealignDNAttributesControl** is present on a request to a master server on which replication of Modify DN operations is enabled, it will be propagated to the replica servers. (See "Modify DN operations and replication" on page 57 for more information about replication of Modify DN operations.)

A few simple examples of the use of the Modify DN operation follow. Each request will be expressed in the format of the ModifyDNRequest defined in RFC 2251, as well as in the corresponding invocation command for the z/VM client utility program LDAPMRDN (**ldapmodrdn**). For more information on LDAPMRDN, see *z/VM: TCP/IP User's Guide*.

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
}


ldapmrdn -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"
```
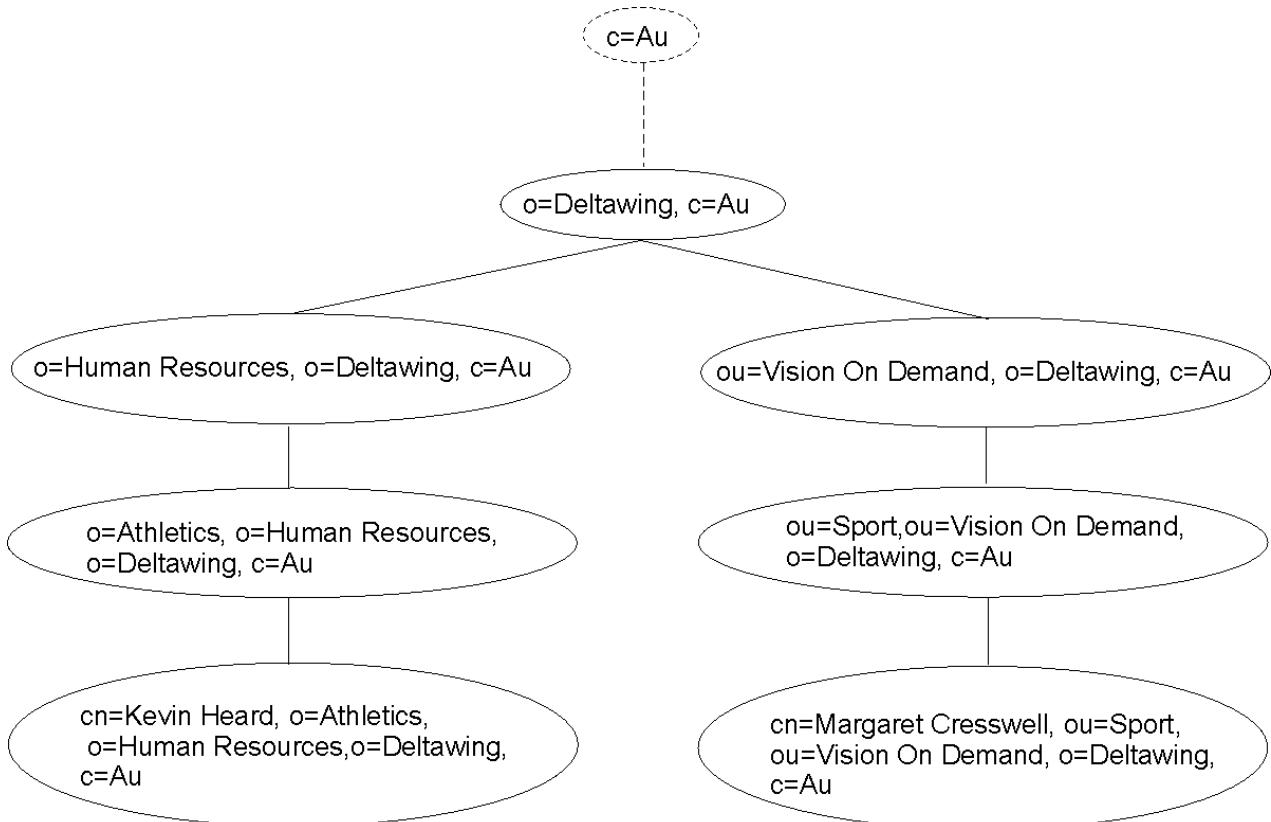

Figure 4. Before Modify DN operation
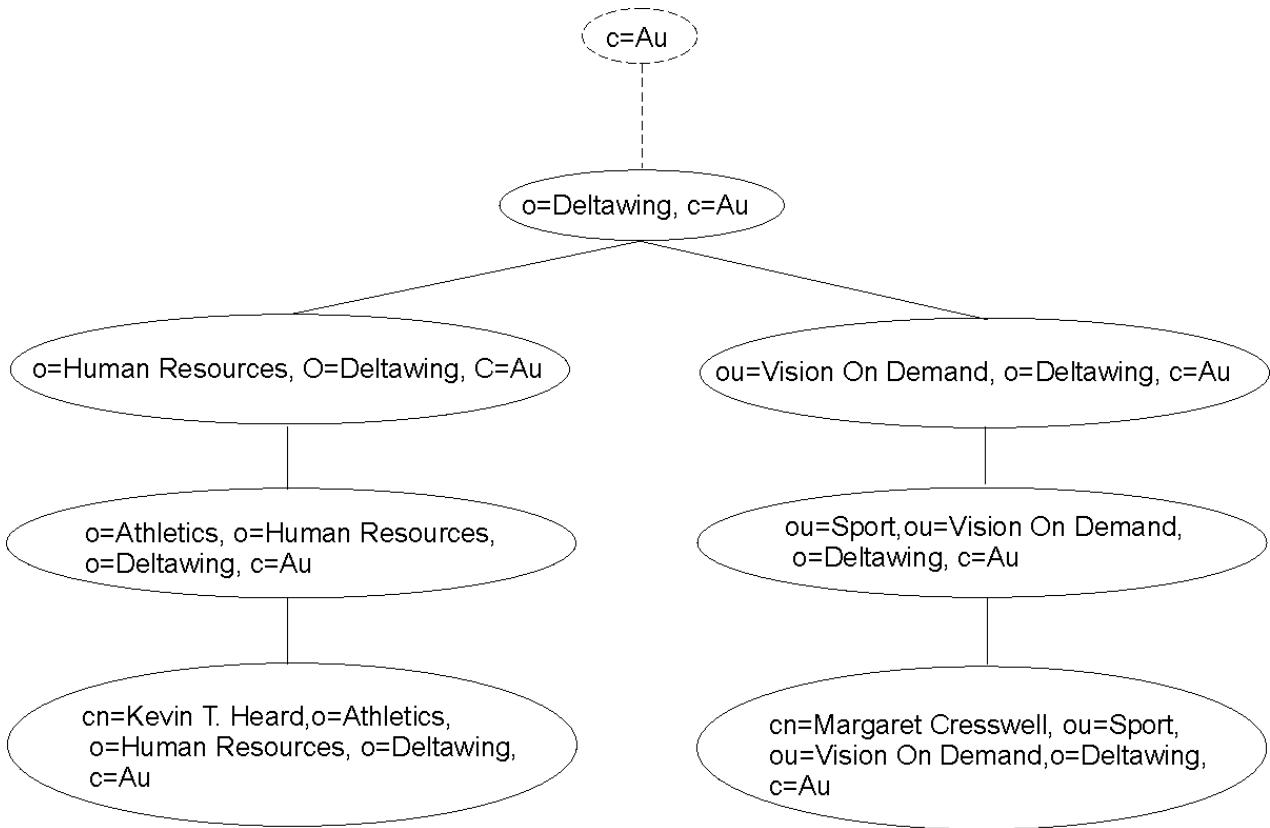
Figure 5. After Modify DN operation

Note: The **-r** parameter specifies that the old RDN attribute value (`cn=Kevin Heard`) will be deleted from the target entry after this operation.

Example 2: Simple Modify DN of non-leaf node

```
ModifyDNRequest ::= {
 entry              o=Athletics, o=Human Resources, o=Deltawing, c=AU
 newrdn             ou=College Athletics Dept.,
 deleteoldrdn       FALSE
}


ldapmrdn -h ldaphost -p ldapport -D binddn -w passwd "o=Athletics,
 o=Human Resources, o=Deltawing, c=AU" "ou=College Athletics Dept."
```
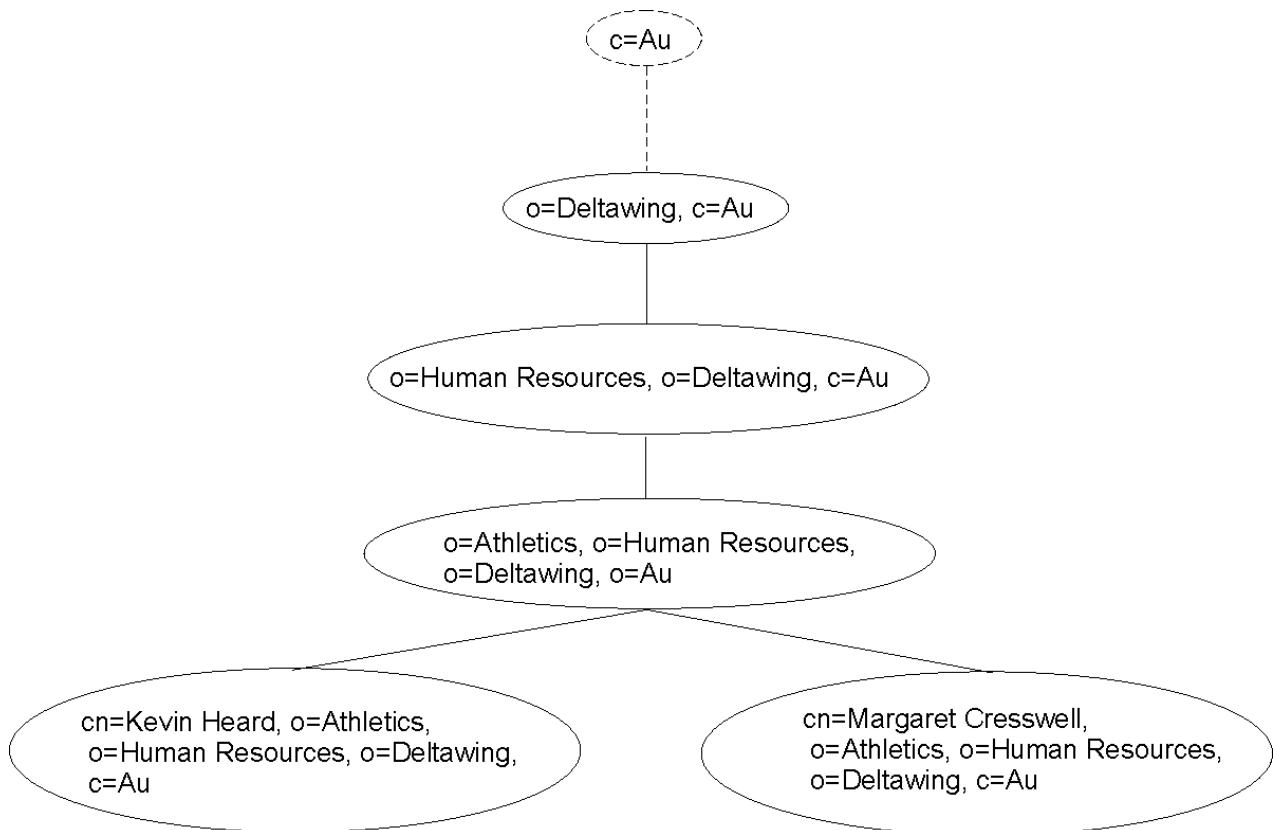
Figure 6. Before Modify DN operation
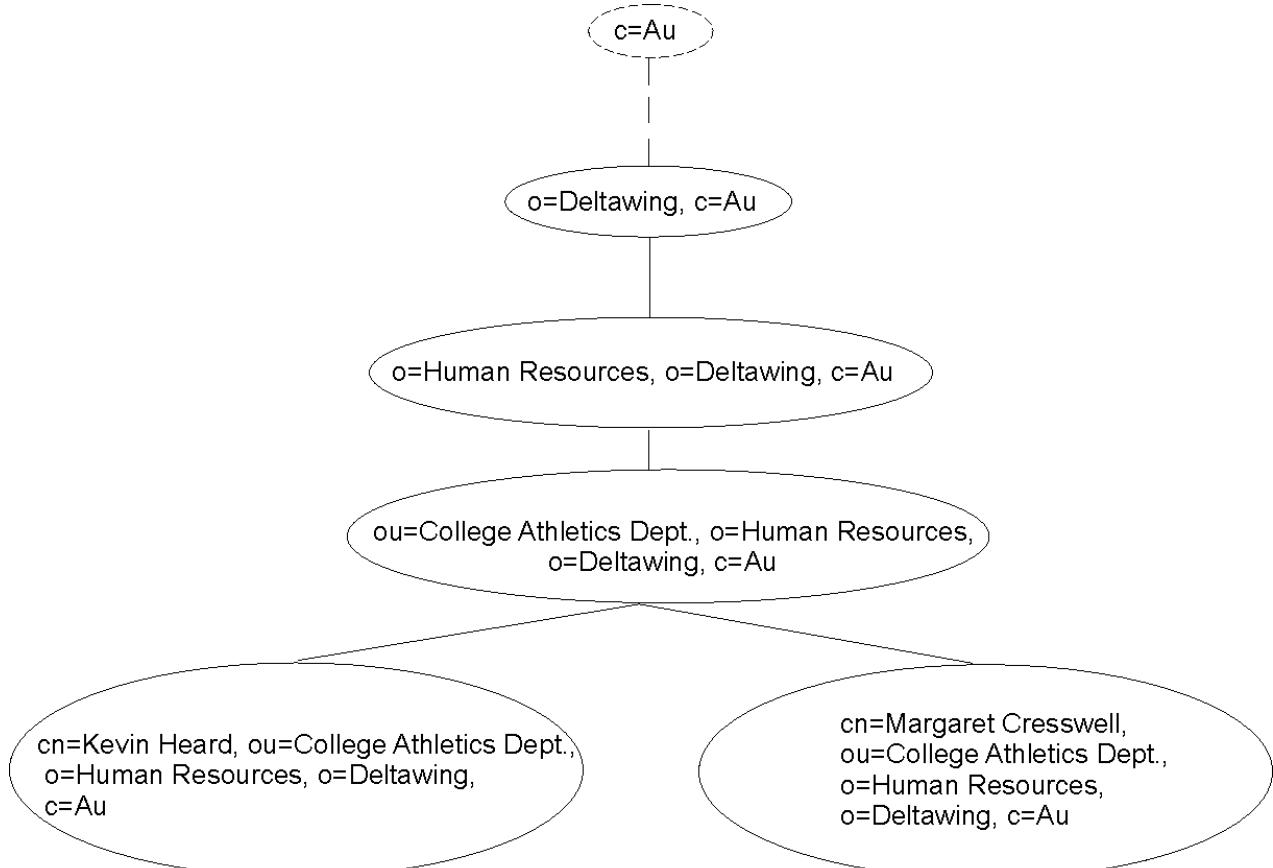
Figure 7. After Modify DN operation

Note: The absence of the **-r** parameter specifies that the old RDN attribute value (o=`Athletics`) will be preserved in the target entry after this operation.

Example 3: Modify DN of non-leaf node with relocation (*newSuperior*)

```
ModifyDNRequest ::= {
entry            o=Athletics, o=Human Resources, o=Deltawing, c=AU
newrdn           o=Adult Athletics
deleteoldrdn     FALSE,
newSuperior      ou=Sport, ou=Vision On Demand, o=Deltawing, o=AU
}


ldapmrdn -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"
```
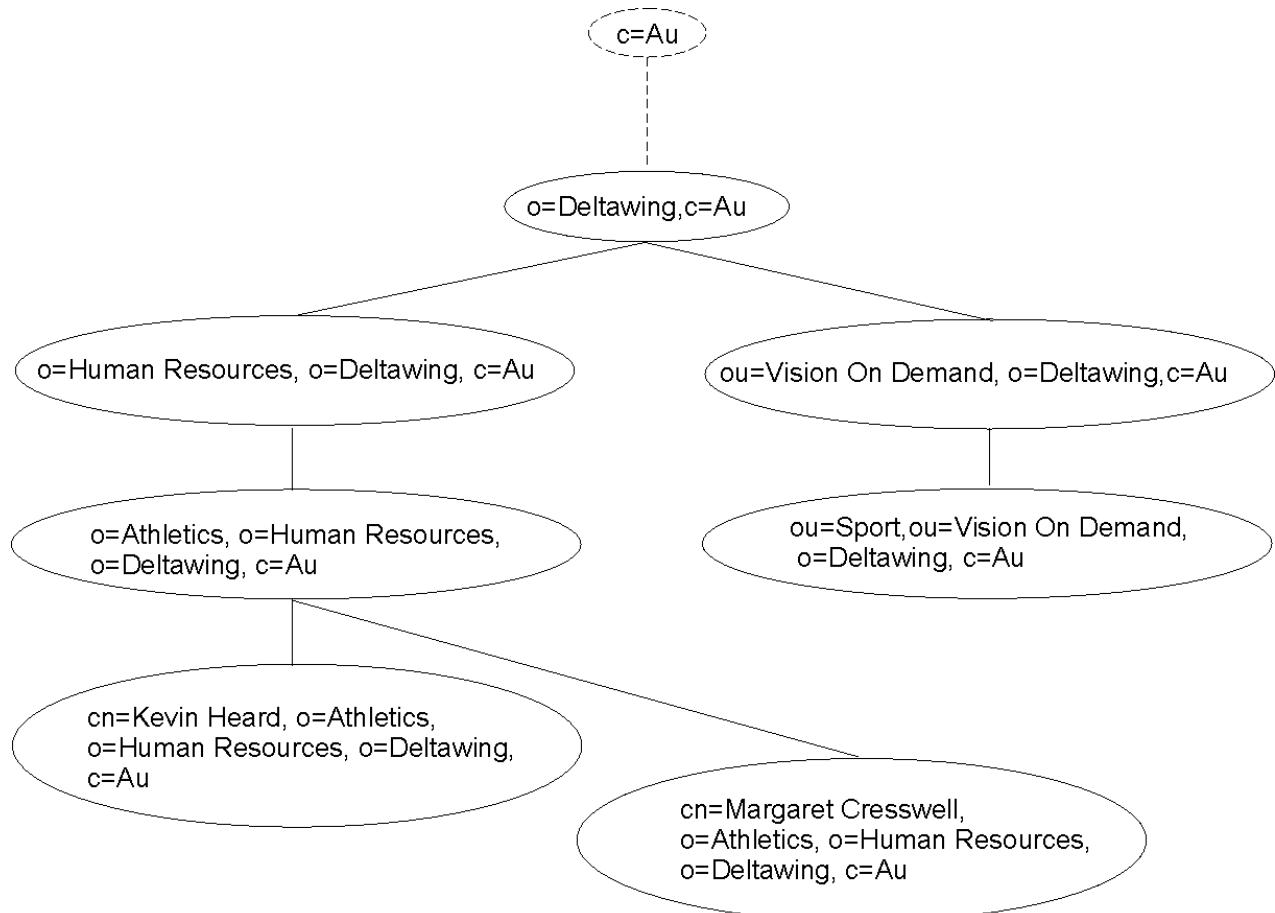

*Figure 8. Before Modify DN operation*
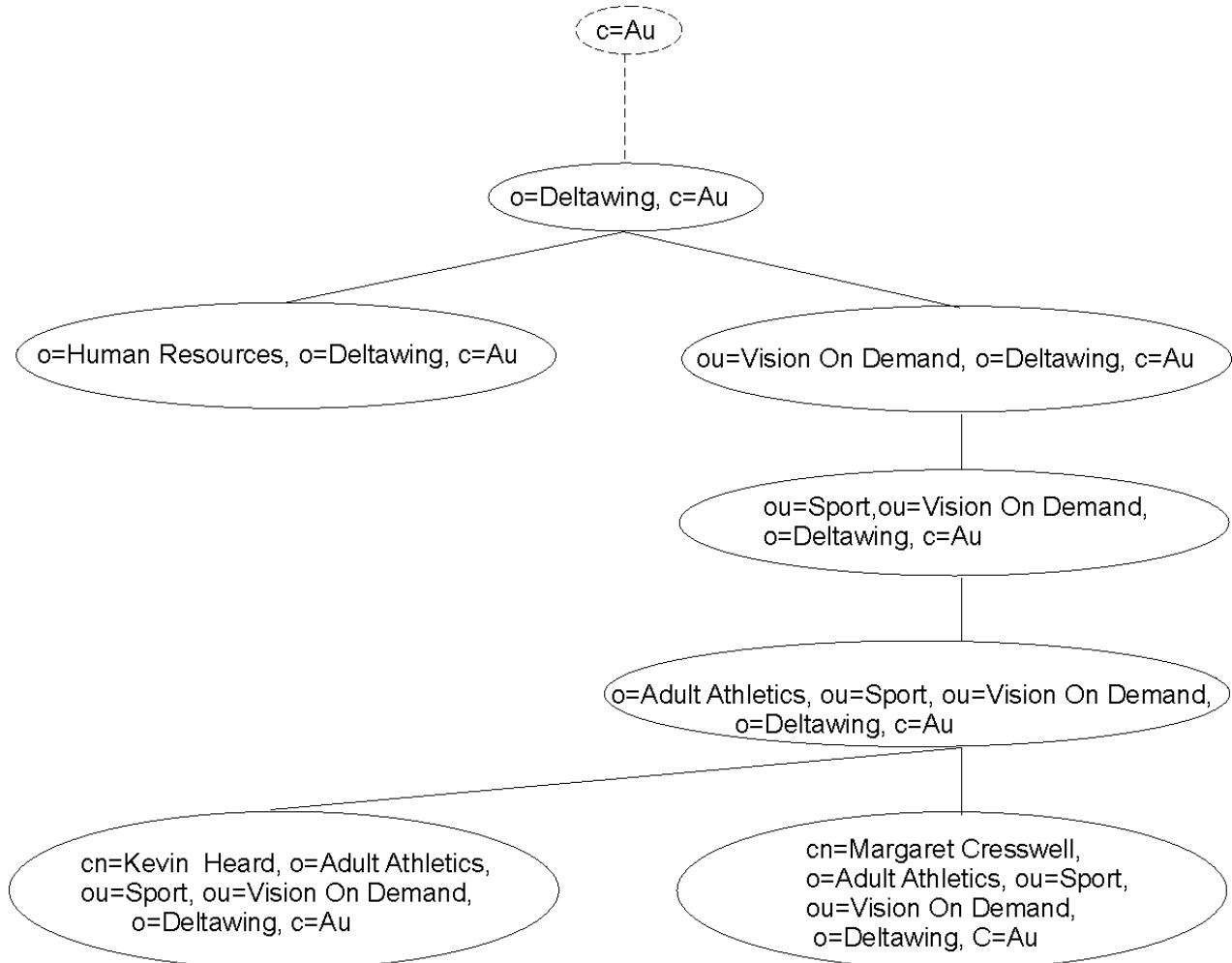
Figure 9. After Modify DN operation

Note: The absence of the **-r** parameter specifies that the old RDN attribute value (o=`Athletics`) will be preserved in the target entry after this operation. The target entry and descendants in its subtree will be relocated in the directory hierarchy.

Considerations in the use of Modify DN operations

As this operation has the potential to significantly change directory data and how it can be accessed, it is important that the user fully understand the data before using the Modify DN operation. Specifically, the user needs to know that:

- The ability of this operation to move directory subtrees has the potential for affecting many entries in the directory in a single operation.
- Certain options may result in modification of additional directory entries which are outside the scope of the directory subtrees being moved. This topic will explain and give examples of how that can occur.
- Because the changes performed to the directory as a result of the operation are committed as a single transaction (or reversed if an error occurs), it may result in a long-running transaction, which may reduce concurrency of other LDAP operations targeted for the same directory entries. See "Concurrency considerations between Modify DN operations and other LDAP operations" on page 44 for more information.
- The scope of the changes may result in unanticipated effects in the directory and may affect user access to these entries. See "Access control changes" on page 46 for more information.
- There are limitations to which directory entries are eligible for the Modify DN operation. See "Eligibility of entries for rename" on page 43 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 DS2LDIF (the **ds2ldif** utility program). For more information about DS2LDIF, see *z/VM: TCP/IP Planning and Customization*. In addition to backing up the directory contents, activity logging should be enabled before nontrivial changes are made to the directory.
- There are considerations if the data to be modified by this operation is being replicated. See "Modify DN operations and replication" on page 57 for more information.

Eligibility of entries for rename

Entries in the directory which are targeted to be renamed in a single Modify DN operation are subject to these constraints:

1. All entries to be renamed must be located in the same LDBM or CDBM backend targeted by the Modify DN operation. The Modify DN operation with *newSuperior* option will move subtree entries within the same LDBM or CDBM backend and will 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 already exist in the backend.

2. Referral entries may be renamed as part of a Modify DN operation. If a referral entry is renamed as part of a Modify DN operation, its corresponding entry in 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 which refer to any of the entries whose DNs were modified in the local directory by a Modify DN operation will need to be manually updated to reflect the changes; no automatic updates are propagated to those servers from the local one.

3. The LDAP server schema entry can not 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 will be 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 may be renamed only if all access control requirements are satisfied for the bound user, as determined by the effective ACL and ownership permissions for those entries and attributes. See "Access control and ownership" on page 44 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 as long as 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 may further result in modification of additional directory entries which are outside the scope of the directory subtrees being renamed or moved.

Changes to all entries affected by the operation are committed at the same time. While modified entries are awaiting the transaction commit point, database locks are held which prevent other concurrent operations from sharing and modifying the data. If many entries undergo modification with this operation, it may result in a long-running transaction which has potential for reducing concurrency of other operations targeted for the same directory entries.

Although the LDAP server is capable of processing concurrent LDAP operations targeted at a given LDBM or CDBM backend while the Modify DN operation is in progress, the extent to which such concurrency is possible will depend on what data in the directory may be needed and locked by the competing operations.

Access control and ownership

For all entries being renamed, the caller must have **w**(rite) permissions for the attribute values that will have to change in all affected entries. In addition, if the *newSuperior* parameter is present on the Modify DN request, the caller must have permissions of **object:a** on the *newSuperior* entry and **object:d** on the target entry at the top of the subtree of entries being moved. If the caller lacks one or more of these permissions, the operation is denied. No access control checking is done against any of the target entry's subordinates even though their DN is changed. It should be noted that if the caller is an effective owner of any of the entries being renamed, the permissions are automatically satisfied for those entries.

In addition, if the **IBMModifyDNRealignDNAttributesControl** accompanies a Modify DN request, then the bound DN must have **w**(rite) permission to all of the attributes that are changed as a result of realignment of the DN values.

Example:

Assume our sample directory contains the following entry which will be the target of a Modify DN operation, and which contains explicit ACL information:

```
dn: o=Athletics, o=Human Resources, ou=Delta Home Media Ltd., o=Deltawing, c=AU
aclEntry: access-id: cn=Mark Crawford, ou=Human Resources, ou=Delta Home Media Ltd.,
 o=Deltawing,c=AU:normal:rswc:sensitive:rsc:object:d

(other attributes not shown)
```

The directory also contains an entry with DN ou=Production, ou=Vision On Demand,o=Deltawing, c=AU which will be the 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	Access class
reportingOrganization	sensitive
workingOrganization	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:

```
ldapmrdn -h ldaphost -p ldapport -D "cn=Mark Crawford, ou=Human Resources, ou=Delta Home
  Media Ltd., o=Deltawing,c=AU" -w passwd -s "ou=Production, ou=Vision On Demand,
  o=Deltawing, c=AU" "o=Athletics, o=Human Resources, ou=Delta Home Media Ltd., o=Deltawing,
  c=AU" "o=Athletics Division"
```

The **-s** parameter specifying *newSuperior* is present on this operation request, so in addition to the access permissions needed for all Modify DN operations (**w** on affected attributes), the user also needs **object:d** on the target entry and **object:a** on the newSuperior entry. The bound user is in the **aclEntry** for the target entry as well as in the **aclEntry** for the newSuperior entry, and has all required access permissions (can write attributes and delete the target entry, and can add objects under the newSuperior entry), so the operation is permitted.

Relocating an entry with DN realignment requested

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):

```
ldapmrdn -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 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 may 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 will be modified. Therefore, before such a move or rename users should carefully consider how ownership and accessibility to entries protected by these attributes may change after the move, and what ACL and ownership changes may be desired, if any.

The following is an example of how a Modify DN operation might affect access controls:

```
ModifyDNRequest ::= {
entry          o=Athletics, o=Human Resources, o=Deltawing, c=AU
newrdn         o=Adult Athletics
deleteoldrdn   FALSE,
newSuperior    ou=Sport, ou=Vision On Demand, o=Deltawing,c=AU
}


ldapmrdn -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"
```
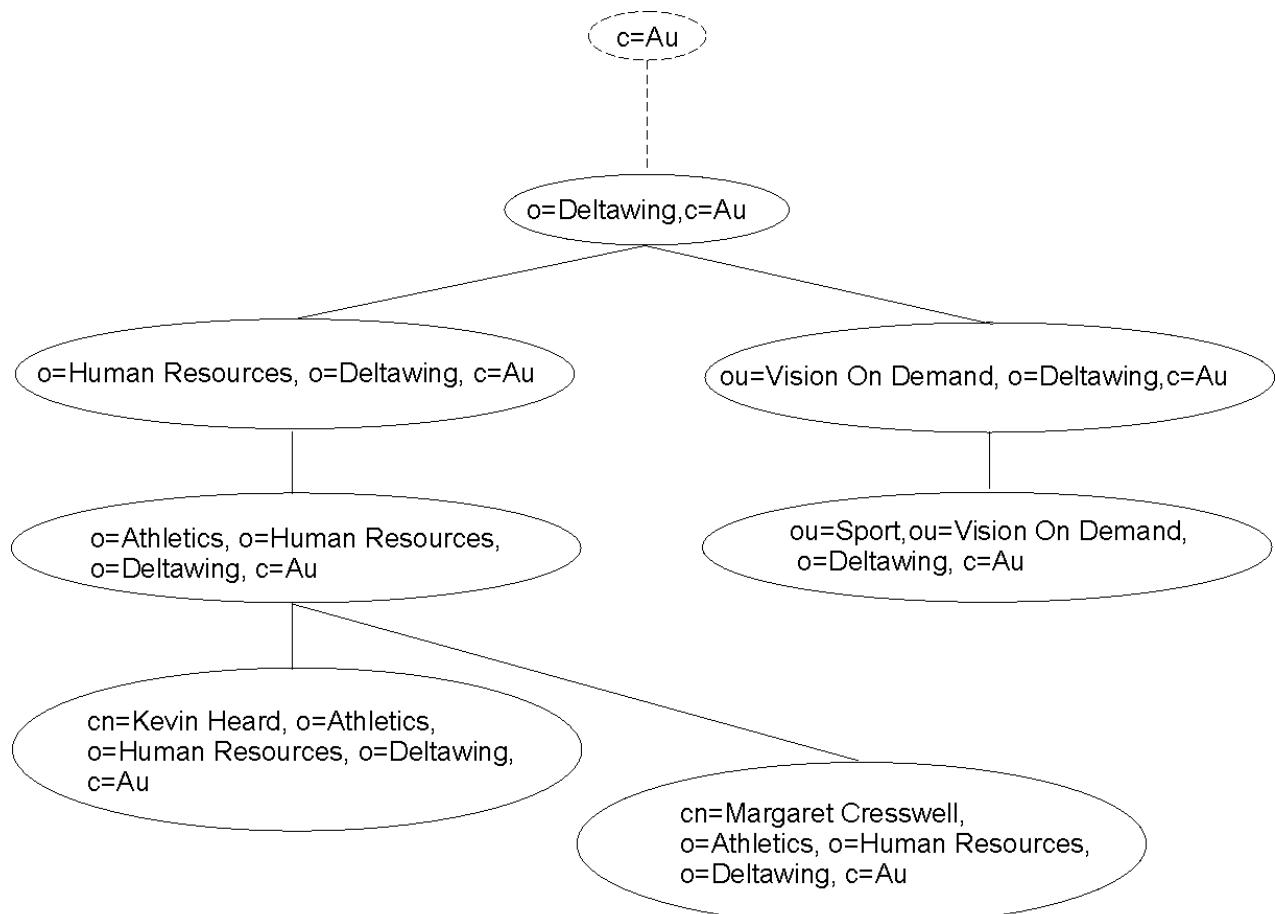

Figure 10. Before Modify Dn operation
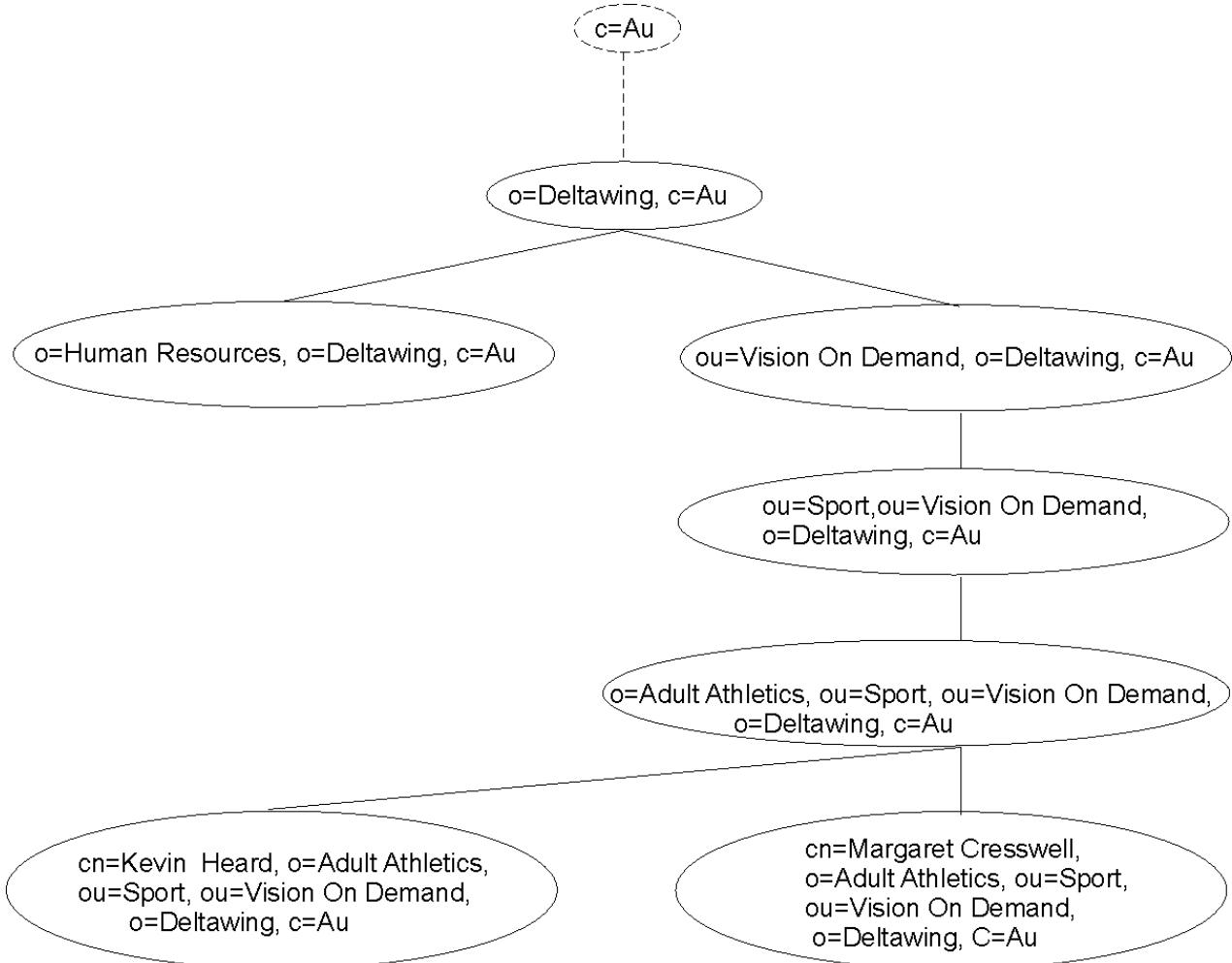
Figure 11. After Modify DN operation

Assume that the entry with DN o=Human Resources, o=Deltawing, c=AU has an explicit propagating ACL containing the following **aclEntry**:

```
aclEntry: access-id: cn=Mark Edmondson, ou=Vision On Demand, ou=Delta Home Media
Ltd.,
o=Deltawing,c=au:normal:rwcs:sensitive:rwcs:critical:rws:object:d
```

Also, assume that the entry with DN ou=Sport, ou=Vision On Demand, o=Deltawing, c=AU has an explicit propagating ACL containing the following **aclEntry**:

```
aclEntry: access-id: cn=Mark Edmondson, ou=Vision On Demand, ou=Delta Home Media
Ltd.,
 o=Deltawing,c=au:normal:rws:sensitive:r:critical:r:object:a
```

If the user bound as DN cn=Mark Edmondson, ou=Vision On Demand, ou=Delta Home Media Ltd., o=Deltawing,c=AU performs the example Modify DN operation, there are at least two consequences which should be noted:

- While this DN previously had **rwcs** permissions on sensitive attributes in the entry o=Athletics, o=Human Resources, o=Deltawing, c=AU and **rws** permissions on critical attributes in the same entry, this DN has only **r** access on both sensitive and critical attributes in the entry after the relocation. It might be expected that a given DN will have the same accessibility to specific entries and data in the directory after a Modify DN operation as it had to those entries and data before the operation, but this example demonstrates that such an expectation is not valid.

- If, after completion of the Modify DN operation, the bound user decides that they want to return the moved entry (and its subordinates) back to their original location in the directory hierarchy, this will not be possible with the access controls currently in place. The bound DN has only **object:d** permission on the old superior node (`"o=Human Resources, o=Deltawing, c=AU"`) where **object:a** is needed to effect the move of an entry or subtree under the superior node, and the bound DN has only **object:a** permission on the moved entry (`"o=Adult Athletics, ou=Sport, ou=Vision On Demand, O=Deltawing, c=AU"`) where **object:d** is needed to move the entry. Therefore, while it may be expected that a given DN can reverse a Modify DN operation under all circumstances, this example demonstrates that such an expectation is not valid.

Ownership changes

When the *newSuperior* parameter accompanies the Modify DN request, any entries in a relocated subtree which had explicit owners before the relocation will preserve that explicit ownership after the relocation has been performed. Any entries in the relocated subtree which inherited ownership before relocation will continue to inherit ownership following relocation. If the owning entry before relocation was a node superior to the relocated entry, the owning entry will be the new superior entry. If the owning entry was an entry within the relocated subtree, the owning entry is preserved following the relocation.

Any entries in the relocated subtree which propagated ownership to subordinates before relocation continue to propagate ownership to subordinates after the relocation.

Refer to the example in "Access control changes" on page 46.

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 may change the effective owner of that entry and of its subordinates.

Modify DN operations related to suffix DNs

The Modify DN operation can be used to modify the DNs of any and all entries in an 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 may also be renamed. Suffix entries may be renamed to become non-suffix entries and suffix entries may be renamed such that they continue to be suffix entries. In addition, non-suffix entries may be renamed to become suffix entries. This section provides example scenarios for rename operations which involve suffix entries. It summarizes constraints which have been adopted for the LDAP directory implementation which are not defined in the protocol behavior prescribed by RFC 2251 for the Modify DN operation. Examples are provided on how various renaming scenarios may be accomplished, and factors to be considered when performing these operations are discussed.

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 will apply which are not defined by RFC 2251 in the description of the protocol behavior:

1. If an entry being renamed will become (or remain) a suffix, the new DN must be designated in the server's configuration file as a suffix for the backend, otherwise the operation will not be permitted.

2. The *newRdn* parameter of the Modify DN request must contain a non-null value, otherwise the operation request will be treated as an error.

3. If the *newSuperior* parameter is present, it may contain a zero-length string signifying that the 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:

c=us

o=MyCompany,c=us

suffix: ou=End_GPL, o=MyCompany, c=us

suffix ou=Endicott, o=MyCompany, c=us

ou=Endicott,o=MyCompany, c=us

ou=End_GPL,o=MyCompany, c=us

ou=s390_sw,ou=End_GPL,o=MyCompany, c=us

ou=s390_sw, ou=Endicott,o=MyCompany,c=us

cn=Li,ou=s390_sw,ou=End_GPL,o=MyCompany, c=us

cn=Li,ou=s390_sw,ou=Endicott,o=MyCompany, c=us
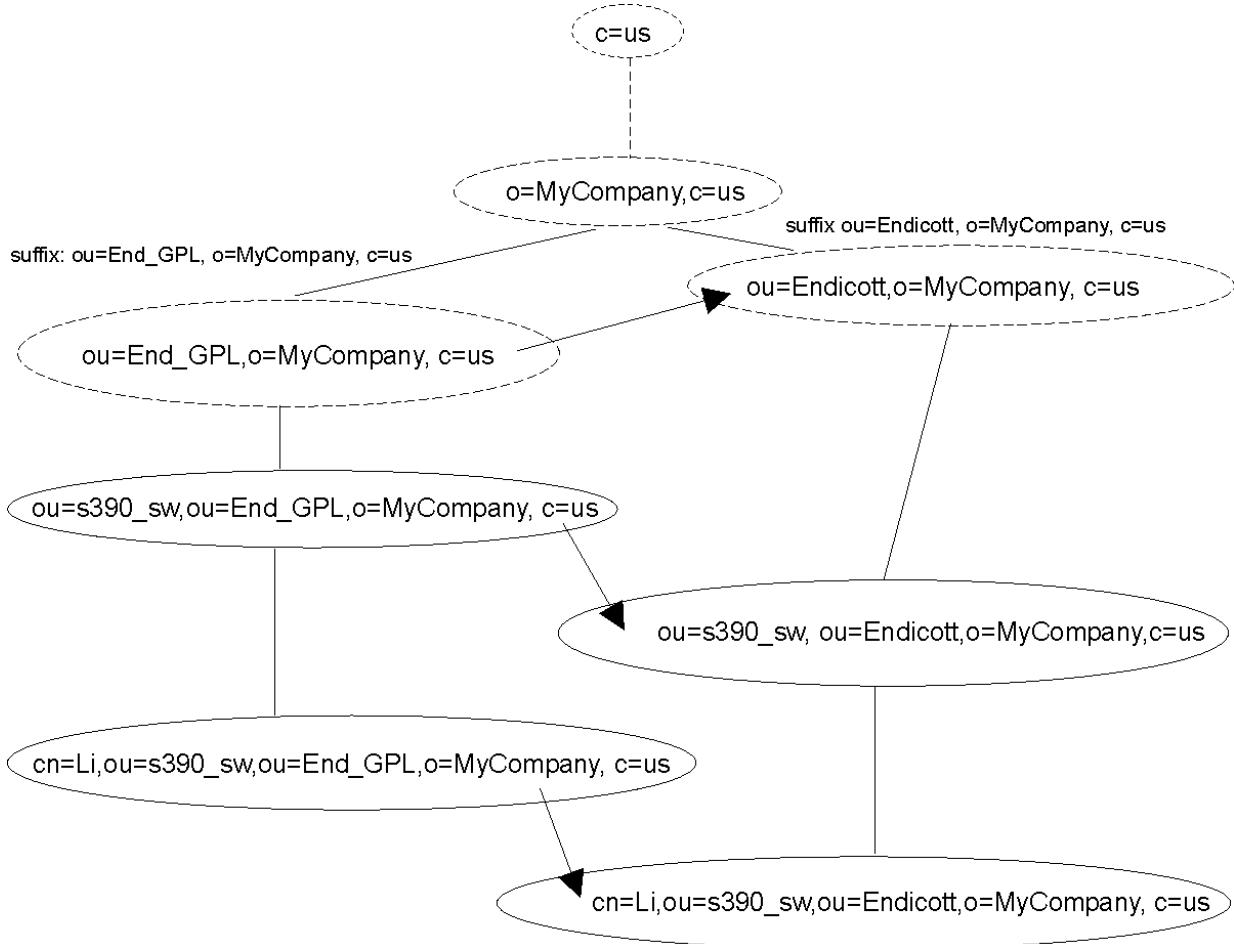
Figure 12. Suffix rename with no new superior

The new DN must be already designated as a suffix for this backend, otherwise this operation will fail.

The operation is performed the same as a rename of any other RDN in the directory

  a. Send Modify DN operation request with
     target=ou=End_GPL, o=MyCompany, c=US
     newRdn=ou=Endicott

This results in renaming ou=End_GPL, o=MyCompany, c=US to ou=Endicott, o=MyCompany, c=US and in renaming subordinate entries accordingly.

2. Rename of suffix DN with an accompanying *newSuperior*, and the new DN remains a suffix after the rename is completed. 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:
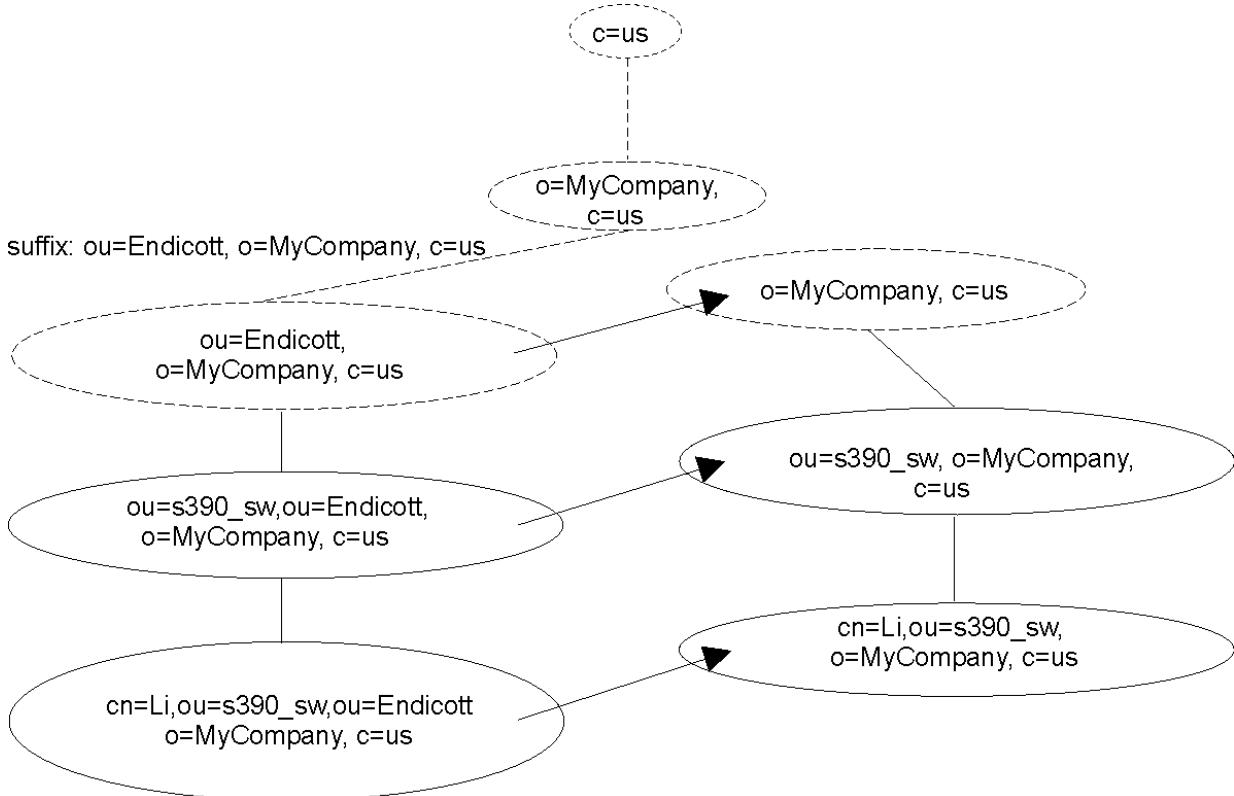
Figure 13. Suffix rename with new superior

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 desired target suffix without a resulting conflict from overlapping suffixes.

c. Send a Modify DN operation request with:

   target= `o=OurTemporarySuffix`
   newRdn= `o=MyCompany`
   newSuperior= `c=us`

This step results in renaming the temporary suffix o=OurTemporarySuffix to the desired suffix
o=MyCompany, c=us, thereby accomplishing the rename from ou=Endicott, o=MyCompany,
c=us to o=MyCompany, c=us. In the process, subordinate entries would be renamed
accordingly.

3. This example shows the renaming of a suffix to another overlapping suffix higher in the directory
   hierarchy. A similar scenario could also be performed involving the rename of a suffix to another
   overlapping suffix, where the new name is a suffix lower in the directory hierarchy. 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:
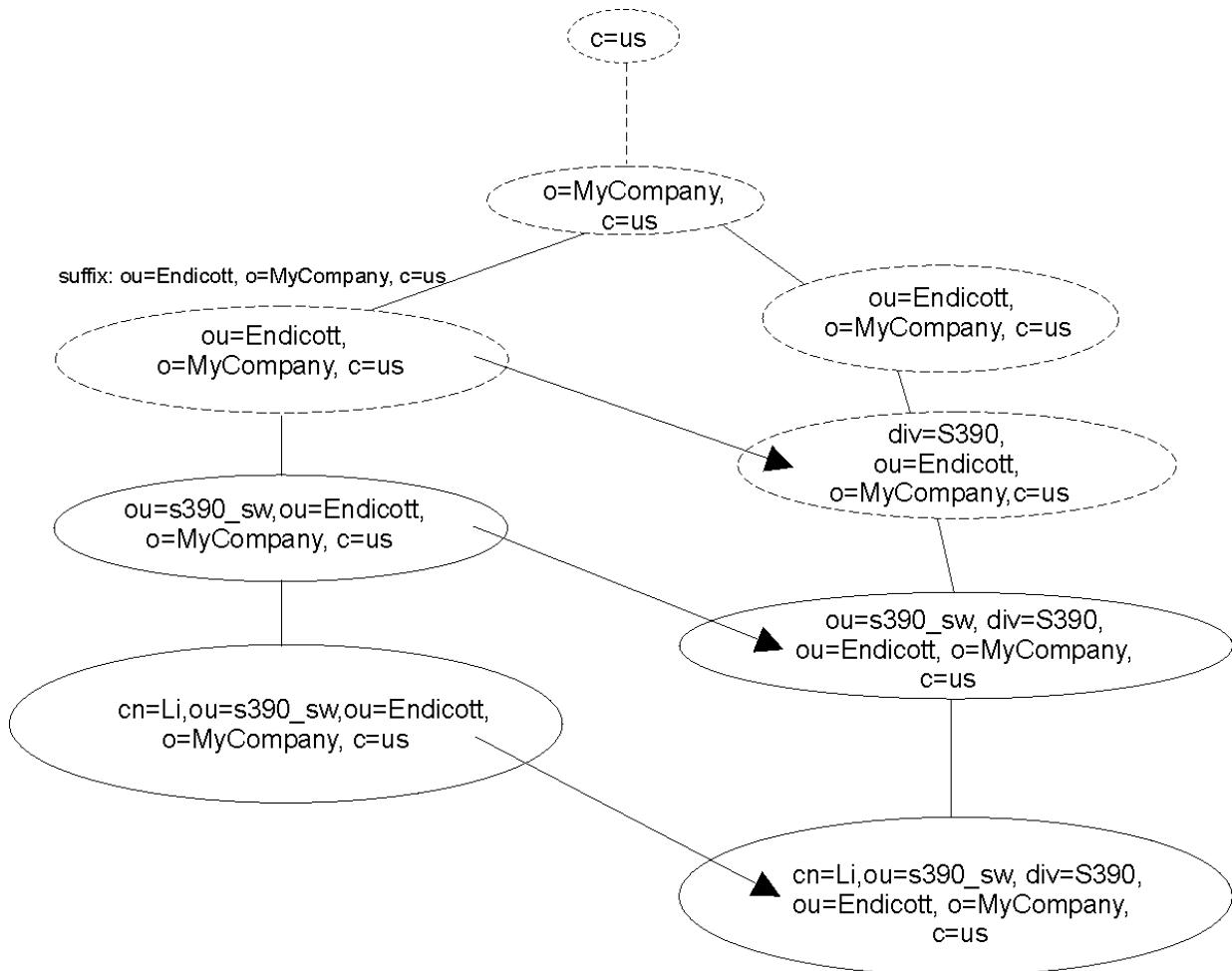
Figure 14. Overlapping suffix rename A

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 that 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(s), add the temporary suffix (`o=OurTemporarySuffix` in our examples), restart the replica server(s).

b. On the master server, perform the previous Steps and from the examples above. This will result in the intermediate rename to be performed on the master server and the results to be propagated to the replica server(s).

c. Stop the replica server(s), delete the original suffix (`ou=Endicott, o=MyCompany, c=us` in both examples above), add the new suffix (`o=MyCompany, c=us` in the first example above, `div=S390, ou=Endicott, o=MyCompany, c=us` in the second example above), and restart the replica server(s).

d. On the master server, perform the previous Step from the examples above. This will result in the rename of entries to the final destination on the master server and in the results being propagated to the replica server(s).

4. Rename of suffix DN (some component other than RDN), and the new DN remains a suffix after the rename is completed. 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:
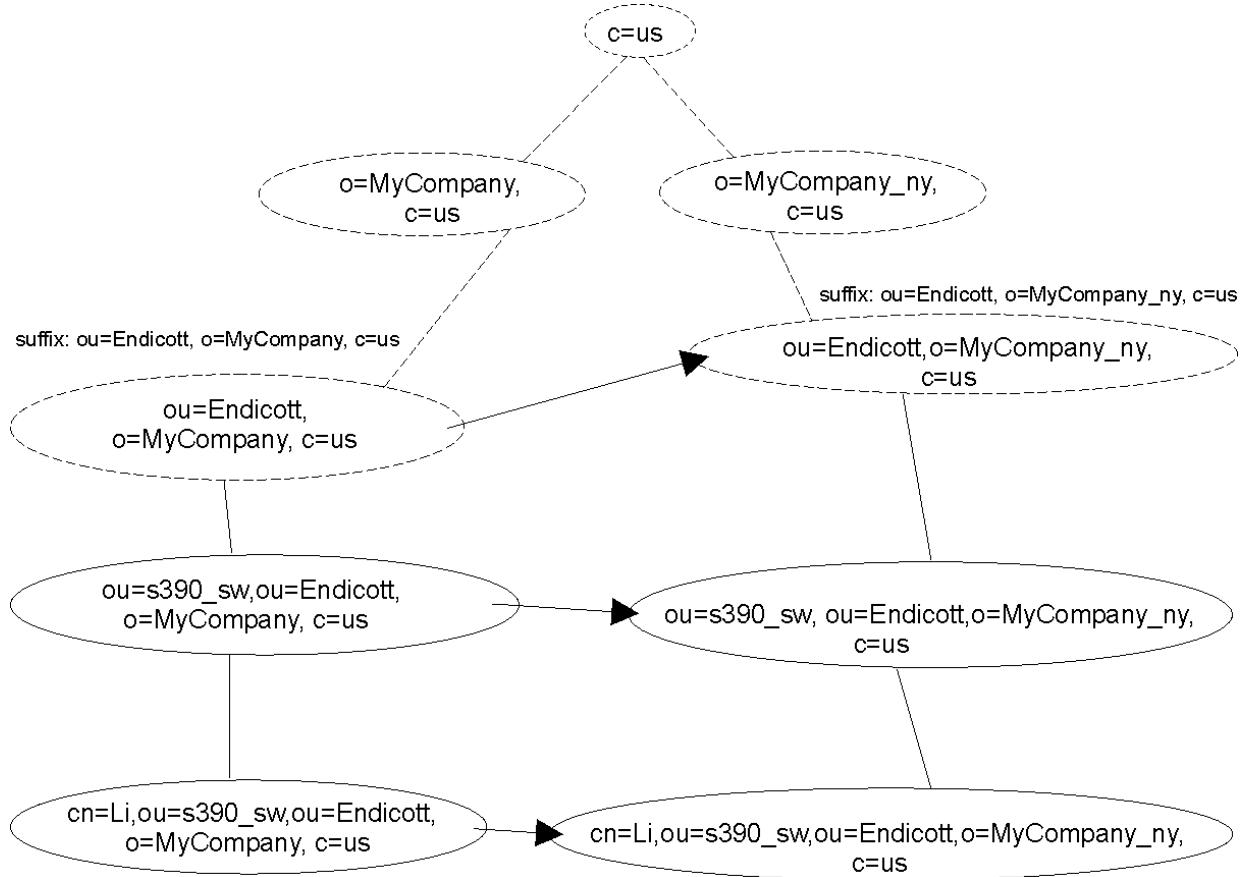
Figure 15. Overlapping suffix rename B

The new DN must be already designated as a suffix for this backend, otherwise this operation will fail. The operation is performed the same as a rename of any other DN in the directory. The product will permit the rename to occur in one step, even if an entry for *newSuperior* does not already exist, since the newly-named entry will become a suffix entry.

  a. Send a Modify DN operation request with
     target= `ou=Endicott, o=MyCompany, c=us`
     newRdn= `ou=Endicott`
     newSuperior= `o=MyCompany_ny, c=us`

     This results in renaming the DN from `ou=Endicott, o=MyCompany, c=us` to `u=Endicott, o=MyCompany_ny, c=us` and in renaming subordinate entries accordingly.

5. Rename of suffix DN (including some component other than RDN), with an accompanying *newSuperior*, but the new DN is no longer a suffix. 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:
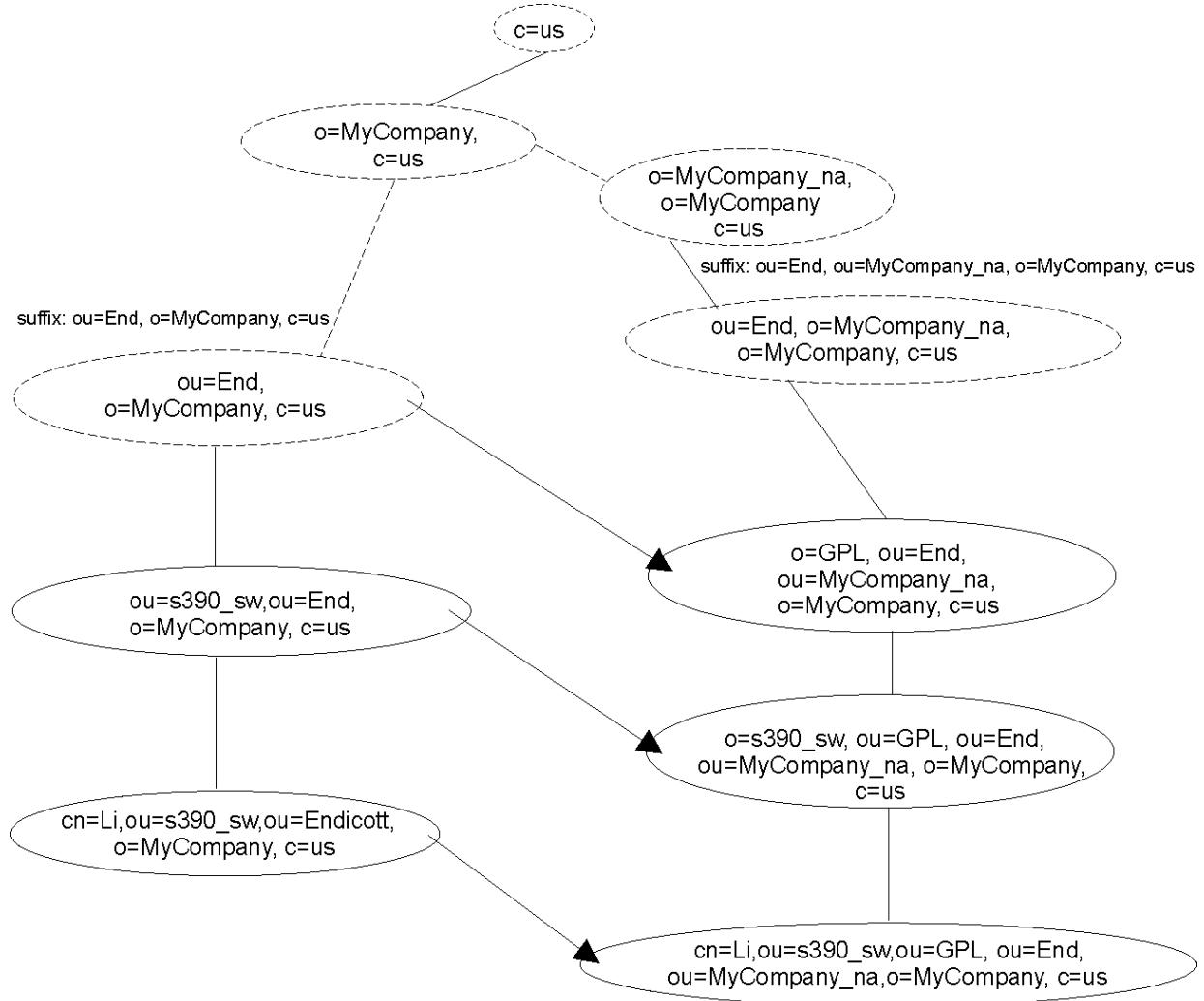
Figure 16. Suffix rename to non-suffix entry

The *newSuperior* entry must already exist before this operation will be 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:
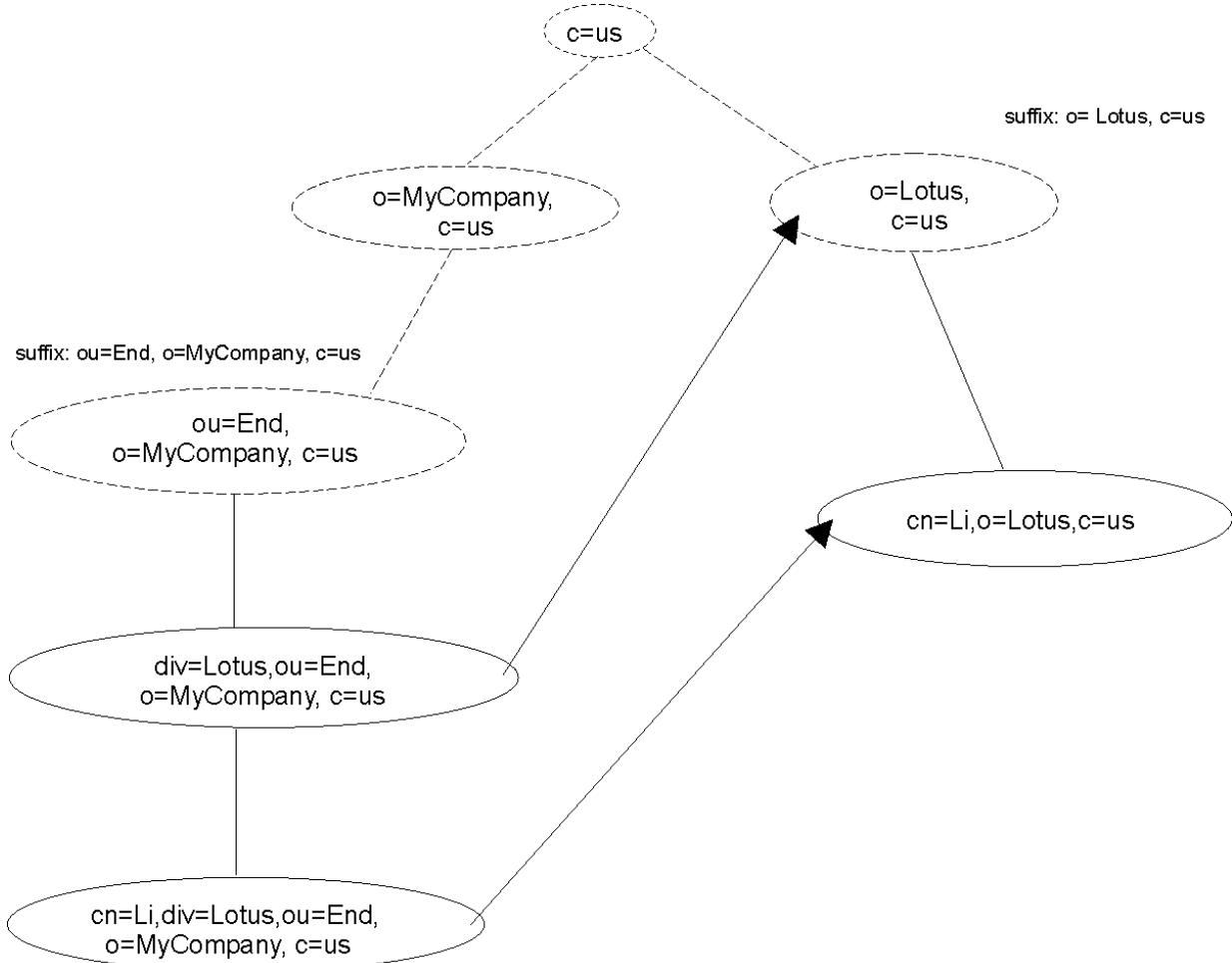
Figure 17. Rename non-suffix entry to suffix entry

The new DN must be already designated as a suffix for this backend, otherwise this operation will fail.

a. Send a Modify DN operation request with
   target= `div=Lotus, ou=Endicott, o=MyCompany, c=us`
   newRdn= `o=Lotus`
   newSuperior= `c=us`

This step results in renaming `div=Lotus, ou=Endicott, o=MyCompany, c=us` to `o=Lotus, c=us` and in renaming subordinate entries accordingly.

Modify DN operations and replication

Modify DN operations may be classified into two categories:

1. Simple Modify DN operations are those 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 LDBM or CDBM backend where a Modify DN operation is applied.

If advanced replication is configured, simple Modify DN operations are accepted only 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 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 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 performed only in a basic replication environment and not in an advanced replication environment.

The master server might 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 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 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 can 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

- The deletion of existing replica entries
- The modification of existing replica entries in the 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 need only to be specified if the replica server does not support anonymous binds.

```
ldapsrch -h ldaphost -p ldapport -D binddn -w passwd
 -s base -b "" objectclass=* ibm-enabledCapabilities
```

where *ldaphost* represents the hostname 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 is capable of supporting 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 highly recommended that the LDAP server administrator 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 can not be replicated, there are several means of recovering from this situation. The best method of recovering from this situation is to ensure that all replica servers are reachable from the master server, and that all replica servers are running at a compatible version level (this may entail stopping some replica servers and restarting them at a compatible version level). Once this state has been reached, queued changes awaiting propagation to replica servers will drain from the queue at the master server and the replication process will resume normal operation.

An alternative is to delete the 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 brought up to a compatible server version) it will be necessary to take a backup of the master server contents and restore those contents to the replica server before restarting it, to ensure the two directories are synchronized.

Chapter 5. 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 dataset 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.

Note: While it is possible for the SDBM backend to issue a SETROPTS RACLIST(FACILITY) REFRESH command to refresh SETROPTS options, it is currently not possible for the backend to issue a SETEVENT REFRESH command to activate VMXEVENT profile changes.

For more information about the supported RACF commands, see *z/VM: RACF Security Server Command Language Reference*.

For information about getting your LDAP server configured with SDBM, see Setting up for SDBM in *z/VM: TCP/IP Planning and Customization*.

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 invokes 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" on page 61 for more information.
- LDBM or CDBM native authentication bind: The RACF user ID specified in the native authentication entry is used. For more information, see Native authentication in *z/VM: TCP/IP Planning and Customization*.
- Certificate bind: The RACF user ID associated with the certificate is used.

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 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/VM LDAP server using RACF access support but having a non-RACF security manager succeeds if the __passwd() call made by the LDAP server is successful. However, no group membership information will be available for the bound distinguished name if the security manager is not RACF.

- An LDAP simple bind made to a z/VM LDAP server 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 *errno* values returned by __passwd() has an LDAP reason code defined for them:

Table 7. The errno values returned by _passwd()		
***errno* value**	**Reason**	**Text**
EACCES	R000104	The password is not correct
EINVAL	R000105	A bind argument is not valid
EMVSERR	R004107	The __passwd function failed; not loaded from a program controlled library
EMVSEXPIRE	R000100	The password has expired
EMVSPASSWORD	R000101	The new password is not valid
EMVSSAFEXTRERR	R000102	The user id has been revoked
EMVSSAF2ERR	R000104	The password is not correct
EMVSSAF2ERR (system problem)	R004176	The __passwd() function failed with error *error_code*
EMVSSAF2ERR (userid problem)	R000104	The password is not correct
ESRCH	R000104	The password is not correct or the user id is not completely defined (missing password or uid)

Note: The same reason codes are issued when binding using a password or a password phrase.

The return code returned by LDAP is **LDAP_OPERATIONS_ERROR** when the errno value is EMVSERR or EMVSSAF2ERR (system problem). For the other errno values, the return code is LDAP_INVALID_CREDENTIALS.

Binding with SDBM using password policy

When authenticating with a user in the SDBM backend, the password policy applied is determined by RACF. Therefore, any configured 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 RACF, the SDBM backend supports only the following **PasswordPolicy** response control error codes during bind: **accountLocked, insufficientPasswordQuality, mustSupplyOldPassword**, and **passwordExpired**. See "PasswordPolicy" on page 348 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 LDAP 3 clients. See Appendix B, "Supported server controls," on page 345 for more information.

Note the **authenticateOnly** control is not necessary if there is no 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.

The user, group, and connection profile fields defined by RACF 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 (Table 8 on page 63), group (Table 9 on page 66), and connection (Table 10 on page 66) profiles. 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 8. Mapping of LDAP attribute names to RACF 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, UAUDIT, or any other one-word values, such as NOEXPIRED and NOOVM	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

Table 8. Mapping of LDAP attribute names to RACF fields (user) (continued)

RACF segment name	RACF keyword in altuser/ adduser/listuser	LDAP attribute name
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
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
User OVM segment	FSROOT	racfOvmFileSystemRoot
User OVM segment	HOME	racfOvmHome
User OVM segment	PROGRAM	racfOvmInitialProgram
User OVM segment	UID	racfOvmUid

Note: The following fields are for z/OS® use and it is recommended that these fields not be used for z/VM.

RACF segment name	RACF keyword	LDAP attribute name
CICS® segment	OPCLASS	racfOperatorClass
CICS segment	OPIDENT	racfOperatorIdentification
CICS segment	OPPRTY	racfOperatorPriority
CICS segment	TIMEOUT	racfTerminalTimeout
CICS segment	XRFSOFF	racfOperatorReSignon
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

RACF segment name	RACF keyword	LDAP attribute name
DFP segment - common to group or user	STORCLAS	SAFDfpStorageClass
LANGUAGE segment	PRIMARY	racfPrimaryLanguage
LANGUAGE segment	SECONDARY	racfSecondaryLanguage
OPERPARM segment	ALTGRP	racfAltGroupKeyword
OPERPARM segment	AUTH	racfAuthKeyword
OPERPARM segment	AUTO	racfAutoKeyword
OPERPARM segment	CMDSYS	racfCMDSYSKeyword
OPERPARM segment	DOM	racfDOMKeyword
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
TSO segment	ACCTNUM	SAFAccountNumber
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

RACF segment name	RACF keyword	LDAP attribute name
WORKATTR segment	WABLDG	racfBuilding
WORKATTR segment	WADEPT	racfDepartment
WORKATTR segment	WANAME	racfWorkAttrUserName
WORKATTR segment	WAROOM	racfRoom

Table 9. Mapping of LDAP attribute names to RACF 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
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

Table 10. Mapping of LDAP attribute names to RACF 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 11. Mapping of LDAP attribute names to RACF fixed fields (resource)

RACF segment name	RACF keyword in setropts	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
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

Table 12. 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	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

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 NOOVM'. 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 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 and 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=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. With the exception for the setrops entry, the top entries can only be compared and searched.

The setrops 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 in *z/VM: TCP/IP Planning and Customization*).

Following is a high-level diagram of the RACF backend.
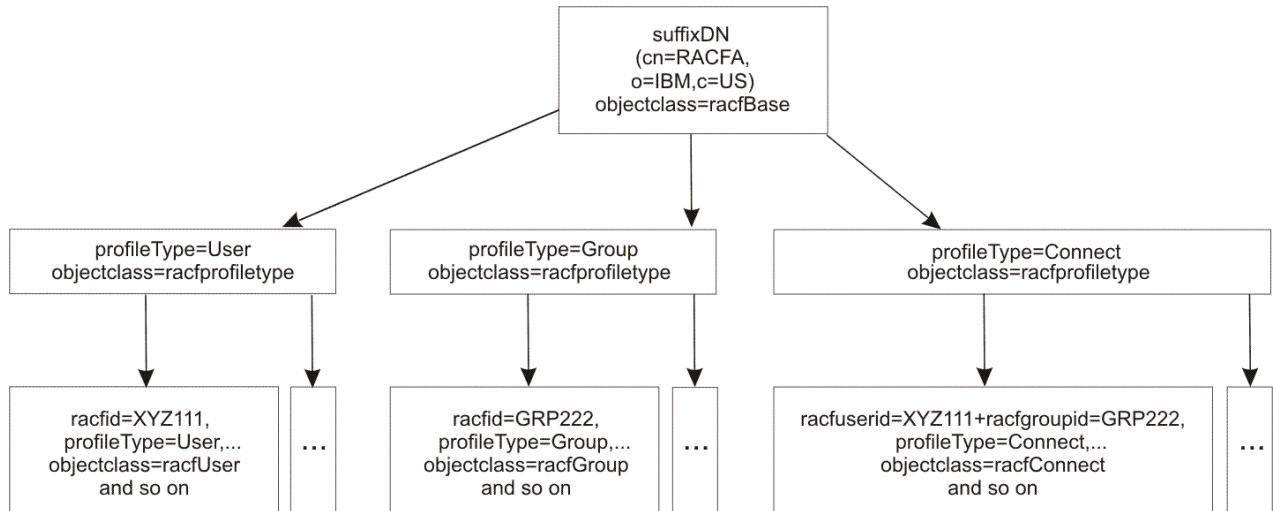
Figure 18. RACF namespace hierarchy (Part 1 of 2)
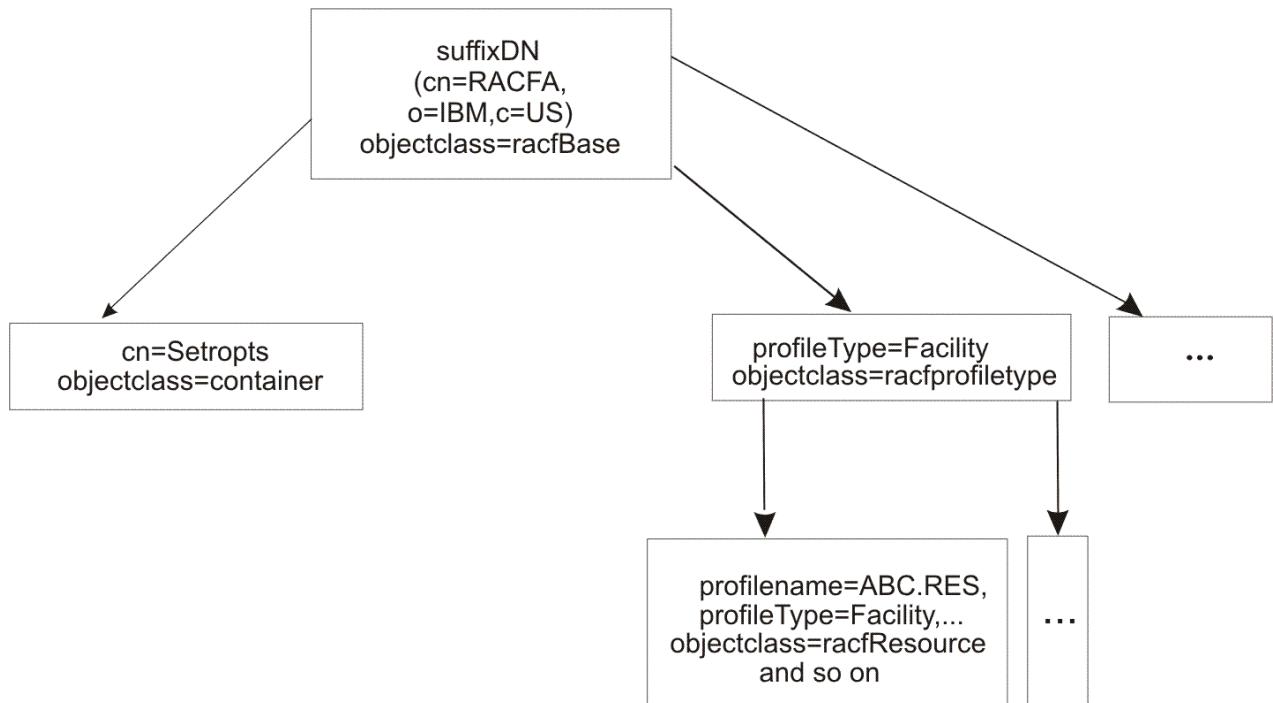
Figure 19. RACF namespace hierarchy (Part 2 of 2)

SDBM schema information

The attributes and object classes used by SDBM to represent RACF values are always in the LDAP server schema.

SDBM support for special characters

An SDBM DN, including the SDBM suffix, can contain the following special characters:

- A plus sign (+), double quote ("), or backslash (\) anywhere in a DN.
- A pound 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=\\#sys#1
```

then the DN for the RACF resource profile a+b in class #x#y would be

```
profilename=a\+b,profiletype=\#x#y,cn=\#sys#1
```

Special characters in a DN returned by SDBM are always escaped by a single backslash. Pound 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=\#sys#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 13 on page 71 and Table 14 on page 79 indicate which commands are issued for various SDBM operations. The RACF commands are issued under the context of the RACF user ID that has 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 searches that result in issuing a RACF **search** command fail and return:

```
ldap_search: Unknown error
ldap_search: additional info: ICH31005I NO ENTRIES MEET SEARCH CRITERIA
```

SDBM operational behavior

Table 13 on page 71 shows how SDBM behaves during different LDAP operations.

Table 13. 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 83 **Bind** Error: No credentials
profiletype=User,*suffixDN*	**Add** Error: Unwilling to perform **Modify** Error: Unwilling to perform **Delete** Error: Unwilling to perform **Modify DN** Error: Unwilling to perform **Compare** Compare attribute **Search base** Return requested attributes **Search one level** See "Searching the entire RACF database" on page 83 **Search subtree** See "Searching the entire RACF database" on page 83 **Bind** Error: No credentials

Table 13. RACF backend behavior (continued)	
Target DN	**LDAP operation behavior**

profiletype=Group,*suffixDN*	**Add**
	    Error: Unwilling to perform
	Modify
	    Error: Unwilling to perform
	Delete
	    Error: Unwilling to perform
	Modify DN
	    Error: Unwilling to perform
	Compare
	    Compare attribute
	Search base
	    Return requested attributes
	Search one level
	    See "Searching the entire RACF database" on page 83
	Search subtree
	    See "Searching the entire RACF database" on page 83
	Bind
	    Error: No credentials

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 83
	Search subtree
	    See "Searching the entire RACF database" on page 83
	Bind
	    Error: No credentials

Table 13. RACF backend behavior (continued)	
Target DN	**LDAP operation behavior**
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 a **setropts extract** RACF command **Search one level** Empty search results (this is a leaf node in the hierarchy) **Search subtree** Perform a **setropts extract** RACF command **Bind** Error: No credentials
profiletype=Connect,*suffixDN*	**Add** Error: Unwilling to perform **Modify** Error: Unwilling to perform **Delete** Error: Unwilling to perform **Modify DN** Error: Unwilling to perform **Compare** Compare attribute **Search base** Return requested attributes **Search one level** See "Searching the entire RACF database" on page 83 **Search subtree** See "Searching the entire RACF database" on page 83 **Bind** Error: No credentials

Table 13. RACF backend behavior (continued)

Target DN	LDAP operation behavior
racfid=XYZ111,profiletype=User, *suffixDN*	**Add** 　　Perform an `adduser` RACF command using USER=XYZ111 **Modify** 　　Perform an `altuser` RACF command using USER=XYZ111 **Delete** 　　Perform a `deluser` RACF command using USER= XYZ111 **Modify DN** 　　Error: Unwilling to perform **Compare** 　　Compare requested attribute with data returned from a profile extract RACF command using USER=XYZ111 **Search base** 　　Perform a profile extract RACF command using USER=XYZ111 **Search one level** 　　Empty search results (this is a leaf node in the hierarchy) **Search subtree** 　　Perform a profile extract RACF command using USER=XYZ111 **Bind** 　　If bind type is not simple, error: Unwilling to perform 　　else use `__passwd()` to verify the user ID and password or password phrase combination and then perform a profile extract RACF command using USER=XYZ111 if gathering group membership
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 a profile extract RACF command using GROUP=GRP222 **Search base** 　　Perform a profile extract RACF command using GROUP=GRP222 **Search one level** 　　Empty search results (this is a leaf node in the hierarchy) **Search subtree** 　　Perform a profile extract RACF command using GROUP=GRP222 **Bind** 　　Error: No credentials

Table 13. RACF backend behavior (continued)

Target DN	LDAP operation behavior
racfuserid=XYZ111+racfgroupid=GRP222, profiletype=Connect,*suffixDN*	**Add** Perform a `connect` RACF command for USER=XYZ111 using GROUP=GRP222 **Modify** Perform a `connect` RACF command for USER=XYZ111 using GROUP=GRP222 **Delete** Perform a `remove` RACF command for USER=XYZ111 using GROUP=GRP222 **Modify DN** Error: Unwilling to perform **Compare** Compare requested attribute with data returned from a profile extract RACF command using USER=XYZ111 **Search base** Perform a profile extract RACF command using USER=XYZ111 **Search one level** Empty search results (this is a leaf node in the hierarchy) **Search subtree** Perform a profile extract RACF command using USER=XYZ111 **Bind** Error: No credentials

Table 13. 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 a profile extract RACF command using PROFILE=ABC.RES in CLASS=FACILITY **Search base** Perform a 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 a 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 have completed a partial update to an entry in RACF. The results will match what occurs if the update were done 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 than normal LDAP support:

- When adding a connection entry that already exists, the entry is modified using the specified attributes. There is no indication returned that the entry already existed.

- When modifying a connection entry that does not exist, the entry is added using the specified attributes. There is no indication returned that the entry did not exist.

Notes about specifying attribute values:

1. There are several SDBM attributes whose value is a RACF user, group name, 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,cn=racfu01,o=ibm,c=us
```

   where `cn=racfu01,o=ibm,c=us` is the SDBM suffix.

   The value returned by SDBM from a search is always the complete LDAP DN.

2. 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.

3. 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 to delete 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.

4. 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)
```

5. 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. Be aware that if you specify multiple add, replace, and delete changes to an attribute in a single modify operation, many LDAPMDFY utilities (including the client z/VM LDAPMDFY) may 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 desired. 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 sort of entries it returns (a complete entry or entries that just 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 14. SDBM search filters	
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 **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:
	
	    user profile extract
	• if scope includes a single group:
	
	    group profile extract
	• if scope includes a single connection:
	
	    connect profile extract
	• if scope includes a single resource:
	
	    resource profile extract
	• if scope includes just the cn=setropts entry:
	
	    setropts extract

Table 14. 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*:
	
	  group profile extract
	• if wildcard in *any_value*:
	  – **search class(group) filter***(any_value)*
	  – followed by **group profile extract** for each group

Filter	Search behavior
Table 14. 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:
	
	```\nsuffix\nprofiletype=user,suffix\nprofiletype=group,suffix\n```
	
	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*)

racfuserid=*any_value*	**Description:**
	    find connection profiles for RACF users whose names match *any_value* (can contain wildcards)
	
	Allowed base:
	
	```\nsuffix\nprofiletype=connect,suffix\n```
	
	Returns:
	    DN-only entries
	
	Commands:
	
	  • if no wildcard in *any_value*:
	
	    user profile extract
	
	  • if wildcard in *any_value*
	
	    – **search class(user) filter**(*any_value*)
	
	    – followed by **user profile extract** for each user

Table 14. SDBM search filters (continued)	
Filter	**Search behavior**

(&(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:
	```\nsuffix\nprofiletype=connect,suffix\n```
	Returns:
	    DN-only entries
	Commands:
	• if no wildcard in *any_value1*:
	    user profile extract
	• if no wildcard in *any_value2*
	    group profile extract
	• if wildcard in both *any_value1* and *any_value2*
	    – **search class(group) filter**(*any_value2*)
	    – followed by **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 actually contained in the group's list of members. As a result, a search of the entry for a universal group does not return most of the group's members. In addition, a search for the connection entry corresponding to a member of a universal group can return different results depending on the connection search filter that is used:

• If the **racfuserid** part of the connection search filter does not contain a wild card, then the connection entry is returned for the specified **racfuserid**.

- If the **racfuserid** part of the connection search filter contains a wild card, then the connection entry for a user is returned only if the user is explicitly contained in the universal group's list of members.

Searching the entire RACF database

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.

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/VM: RACF Security Server Command Language Reference*.

LDAP restriction on RACF data

If a RACF field contains unprintable characters, the value returned in the LDAP output will probably not match the RACF value and will probably not be 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 LDAPSRCH (**ldapsearch**) to retrieve the password or password phrase envelopes, the returned value is base-64 encoded.

Using SDBM to change a user password or password phrase in RACF

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/VM 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 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 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 for RACF user U1 from `abc` to `xyz`, assuming the SDBM suffix is `cn=racfu01,o=ibm,c=us`:

```
ldapsrch -h ldaphost -p ldapport -D racfid=u1,profiletype=user,
  cn=racfu01,o=ibm,c=us  -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 LDAPMDFY 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 `cn=racfu01,o=ibm,c=us`. 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,cn=racfu01,o=ibm,c=us
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:

```
ldapmdfy -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,
  cn=racfu01,o=ibm,c=us -w passwd -f pw.mod
```

   modifies the password or password phrase 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/VM: TCP/IP User's Guide* 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 `cn=racfu01,o=ibm,c=us` is the SDBM suffix.

Example: adding a user to RACF

If the LDIF file `user.add` contains:

```
dn: racfid=newuser,profiletype=user,cn=racfu01,o=hal,c=us
objectclass: racfUser
racfid: newuser
```

The following command adds user ID `newuser` to RACF:

```
ldapadd -h ldaphost -p ldapport -D
racfid=admin1,profiletype=user,cn=racfu01,o=hal,c=us
 -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 an OVM segment for `newuser`, the LDIF file `user.mods` could contain:

```
dn: racfid=newuser,profiletype=user,cn=racfu01,o=hal,C=us
changetype: modify
objectclass: racfUserOvmSegment
racfOvmHome: /home/newuser
racfOvmInitialProgram: /home/newuser/bin/startup
racfOvmUid : 500
```

The command:

```
ldapmdfy -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,
 cn=racfu01,o=hal,c=us -w passwd -f user.mods
```

modifies the RACF user profile for user ID `newuser`, adding an OVM 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:

```
ldapsrch -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,
 cn=racfu01,o=hal,c=us -w passwd -b "racfid=newuser,profiletype=user,cn=racfu01,o=hal,c=us"
 "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,cn=RACFU01,o=HAL,c=US
racfid=NEWUSER
racfauthorizationdate=08/23/24
racfowner=RACFID=OPERATOR,PROFILETYPE=USER,CN=RACFU01,O=HAL,C=US
racfpasswordinterval=30
racfdefaultgroup=RACFID=SYS1,PROFILETYPE=GROUP,CN=RACFU01,O=HAL,C=US
racflogondays=SUNDAY
racflogondays=MONDAY
racflogondays=TUESDAY
racflogondays=WEDNESDAY
racflogondays=THURSDAY
racflogondays=FRIDAY
racflogondays=SATURDAY
racflogontime=ANYTIME
racfconnectgroupname=RACFID=SYS1,PROFILETYPE=GROUP,CN=RACFU01,O=HAL,C=US
racfhavepasswordenvelope=NO
racfattributes=PASSWORD
racfovmuid=500
racfovmhome=/home/newuser
racfovminitialprogram=/home/newuser/bin/startup
objectclass=RACFBASECOMMON
objectclass=RACFUSER
objectclass=RACFUSEROVMSEGMENT
```

Example: searching for a user's password and password phrase envelopes in RACF

The following search returns the **racfPasswordEnvelope** and **racfPassPhraseEnvelope** attributes:

```
ldapsrch -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,cn=racfu01,
o=hal,c=us -w passwd -L -b racfid=newuser,profiletype=user,cn=racfu01,o=hal,c=us
 "objectclass=*" racfpasswordenvelope racfpassphraseenvelope
```

The result returned is:

```
dn: racfid=newuser,profiletype=user,cn=racfu01,o=hal,c=us
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,cn=racfu01,o=hal,c=us
objectclass: racfGroup
racfid: grp222
```

The following command adds group ID grp222 to RACF:

```
ldapadd -h ldaphost -p ldapport -D
racfid=admin1,profiletype=user,cn=racfu01,o=hal,c=us
 -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 very similar to the corresponding commands for a RACF user. See the examples in this section for a RACF user for more information.

Example: connecting a user to a group in RACF

To connect newuser to group grp222, the LDIF file connect.add could contain:

```
dn: racfuserid=newuser+racfgroupid=grp222,profiletype=connect,cn=racfu01,o=hal,c=us
objectclass: racfconnect
racfuserid: newuser
racfgroupid: grp222
```

The command:

```
ldapadd -h ldaphost -p ldapport -D
racfid=admin1,profiletype=user,cn=racfu01,o=hal,c=us
 -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:

```
ldapsrch -h ldaphost -p ldapport -D
racfid=admin1,profiletype=user,cn=racfu01,o=hal,c=us
```

```
 -w passwd -b "racfuserid=newuser+racfgroupid=grp222,profiletype=connect,
  cn=racfu01,o=hal,c=us" "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,cn=racfu01,o=hal,c=us
racfuserid=NEWUSER
racfgroupid=GRP222
racfconnectauthdate=07/18/24
racfconnectowner=RACFID=ADMIN1,PROFILETYPE=USER,cn=racfu01,o=hal,c=us
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:

```
ldapsrch -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,
cn=racfu01,o=hal,c=us
 -w passwd -b "profiletype=connect,cn=racfu01,o=hal,c=us" "racfuserid=newuser"
```

or

```
ldapsrch -h ldaphost -p ldapport -D
racfid=admin1,profiletype=user,cn=racfu01,o=hal,c=us
 -w passwd -b "profiletype=connect,cn=racfu01,o=hal,c=us"
 "(&(racfuserid=newuser)(racfgroupid=*))"
```

For both commands, the results are:

```
racfuserid=NEWUSER+racfgroupid=G1,profiletype=CONNECT,cn=racfu01,o=hal,c=us

racfuserid=NEWUSER+racfgroupid=GRP222,profiletype=CONNECT,cn=racfu01,o=hal,c=us
```

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):

```
ldapdlet -h ldaphost -p ldapport
 -D racfid=admin1,profiletype=user,cn=racfu01,o=hal,c=us -w passwd
 "racfuserid=newuser+racfgroupid=grp222,profiletype=connect,cn=racfu01,o=hal,c=us"
```

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):

```
ldapdlet -h ldaphost -p ldapport
 -D racfid=admin1,profiletype=user,cn=racfu01,o=hal,c=us -w passwd
 "racfid=newuser,profiletype=user,cn=racfu01,o=hal,c=us"
```

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,cn=racfu01,o=hal,c=us
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,cn=racfu01,o=hal,c=us
 -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,cn=racfu01,o=hal,c=us
changetype: modify
racfraclist: facility
racfsetroptsattributes: refresh
```

The command:

```
ldapmdfy -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,cn=racfu01,o=hal,c=us -w
passwd
 -f setropts.mod
```

issues a RACF **setropts** command to refresh the raclisted profiles in the facility class.

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** - NOAUDITOR NOOPERATIONS NOSPECIAL NOUAUDIT
- **racfConnectAttributes** - NOAUDITOR NOOPERATIONS NOSPECIAL
- **racfResourceAttributes** - NOWARNING

Deleting a specific value for these attributes requires that the value itself be specified on the delete operation.

For example, to remove the OPERATIONS and AUDITOR values from the **racfAttributes** values of user ID user1 (leaving any other **racfAttributes** values the user has), issue an LDAPMDFY command with the following file:

```
dn: racfid=user1,profiletype=user,cn=racfu01,o=hal,c=us
changetype: modify
delete: racfAttributes
racfAttributes: OPERATIONS
racfAttributes: AUDITOR
```

To remove all the **racfAttributes** values listed above of user ID user1, issue an LDAPMDFY command with the following file:

```
dn: racfid=user1,profiletype=user,cn=racfu01,o=hal,c=us
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 OVM segment from user ID `user1`, issue an LDAPMDFY command with one of the following files:

```
dn: racfid=user1,profiletype=user,cn=racfu01,o=hal,c=us
changetype: modify
delete: racfAttributes
racfAttributes: OVM
```

or

```
dn: racfid=user1,profiletype=user,cn=racfu01,o=hal,c=us
changetype: modify
add: racfAttributes
racfAttributes: NOOVM
```

Following are some additional examples of deleting attributes:

*
  ```
  dn: racfid=user1,profiletype=user,cn=racfu01,o=hal,c=us
  changetype: modify
  delete: racfProgrammerName
  ```

  Returns: **LDAP_UNWILLING_TO_PERFORM**

  The `racfProgrammerName` attribute is one that cannot be deleted.

*
  ```
  dn: racfid=user1,profiletype=user,cn=racfu01,o=hal,c=us
  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,cn=racfu01,o=hal,c=us
  changetype: modify
  delete: racfBuilding
  ```

  Expected result: successful removal of the attribute **racfBuilding** and **LDAP_SUCCESS** returned.

Chapter 6. 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/VM LDAP server are:

- **Directory data administrator** is allowed to administer all 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 backend.
- **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" on page 91 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/VM LDAP server:

- **Directory data administrator** Users assigned this role have unrestricted access to the entries in the LDBM backend 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 LDBM backend. 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 backend through normal ACL evaluation. ACL filters are still applied, but cannot augment any permissions explicitly denied by this role definition. See Chapter 10, "Using access control," on page 145 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 backend 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 backend. Password policy constraints set by the server do not apply to users assigned the password administrator role. This role has access to the LDBM backend through normal ACL evaluation. ACL filters are still applied, but cannot augment any permissions explicitly denied by this role definition. See Chapter 10, "Using access control," on page 145 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 backend 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 backend (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 10, "Using access control," on page 145 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 backend. 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 backend through normal ACL evaluation. ACL filters are still applied, but cannot augment any permissions explicitly denied by this role definition. See Chapter 10, "Using access control," on page 145 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 backend through normal ACL evaluation. ACL filters are still applied, but cannot augment any permissions explicitly denied by this role definition. See Chapter 10, "Using access control," on page 145 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 CDBM backend must be configured to use the administrative group and roles. 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 in *z/VM: TCP/IP Planning and Customization* 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 91 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 96 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 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 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 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 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`

Table 15. ibm-slapdAdminGroupMember objectclass schema definition (optional and required attributes) (continued)
Attribute description and example

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 91 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 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 16 on page 99 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 CDBM backend configuration and policy entries in *z/VM: TCP/IP Planning and Customization* 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" on page 98 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 Native authentication in *z/VM: TCP/IP Planning and Customization* 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=CDD,o=HAL,c=US
```

and the `cn=User1,ou=CDD,o=HAL,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 on page 99 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 B, "Supported server controls," on page 345 and Appendix C, "Supported extended operations," on page 357 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

Extended operation or control	Directory admin	Replication admin	Schema admin	Server configuration group member	Password admin	No admin	Operational admin	Root admin
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
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

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 on page 100 summarizes the extended operation. See LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization* 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=wire,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=wire,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 7. 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 apply only to entries that have a **userPassword** value stored in a 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 LDBM or CDBM are not subject to LDAP password policy. These users include the LDAP administrator when the password value is specified in the **adminPW** 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 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 or EXTERNAL binds as these authentication mechanisms do not access a password value. LDAP password policy also does not apply to LDBM or CDBM entries participating in native authentication or entries in the SDBM backend. RACF handles the password policy for these users. For more information, see "Binding with SDBM using password policy" on page 62.

During add and modify requests of password values in the 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 at this time, and if the password must be changed at this time for the user.

Password policy entries

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 LDBM and CDBM entries that have a **userPassword** attribute value. The default values specified in Table 18 on page 105 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 18 on page 105 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, or nested group by adding or modifying the single-valued **ibm-pwdGroupPolicyDN** operational attribute value in a group entry. See Chapter 9, "Static, dynamic, and nested groups," on page 133 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 124 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 112 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 18 on page 105. 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 18 on page 105 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 18 on page 105 describes the most restrictive value for each attribute for password policy evaluation. See "Password policy evaluation" on page 112 for more information.

Table 18. 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: <pre>ibm-pwdGroupAndIndividualEnabled: false</pre>
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: <pre>ibm-pwdPolicy: true</pre>
ibm-pwdPolicyStartTime Specifies the time in Zulu format when the 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: <pre>ibm-pwdPolicyStartTime: 20091001170933.867354Z</pre>

Table 18. Password policy attributes and default values (continued)

Attribute description and example

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 113 for more information.

Example:

```
passwordMinAlphaChars:  3
```

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 113 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 113 for more information.

Example:

```
passwordMaxRepeatedChars: 2
```

Table 18. Password policy attributes and default values (continued)

Attribute description and example

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 113 for more information.

Example:

```
passwordMaxConsecutiveRepeatedChars: 2
```

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 113 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
```

Table 18. Password policy attributes and default values (continued)

Attribute description and example

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
```

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
```

Table 18. Password policy attributes and default values (continued)
Attribute description and example
pwdFailureCountInterval Specifies the number of seconds when password failures are removed from the failure counter although no successful authentication has 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`
pwdGraceLoginLimit Specifies the number of times an expired password is used for authentication. If set to 0 and the password has 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 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 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 stored in the **pwdHistory** operational attribute values are used to check new password values 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`

Table 18. Password policy attributes and default values (continued)

Attribute description and example
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`
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`

Table 18. Password policy attributes and default values (continued)

Attribute description and example

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
```

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 113 for more information.

Example:

```
pwdMinLength: 8
```

Table 18. Password policy attributes and default values (continued)
Attribute description and example
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 LDAPCHPW utility is used to change a user's password value. For more information about the LDAPCHPW utility, see *z/VM: TCP/IP User's Guide*.   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 has 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 18 on page 105 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 104 and Table 19 on page 114 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 19 on page 114. The examples in Table 19 on page 114 only display values for the attributes specified in the various groups, not all the default values for the attributes that are not specified.

Table 19. 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

Table 19. Composite group password policy examples (continued)			
Group X password policy	**Group Y password policy**	**Group Z password policy**	**Composite group password policy**
passwordMaxConsecutiveRepeatedChars = 0 passwordMaxRepeatedChars = 5 ibm-pwdPolicy = true	passwordMaxConsecutiveRepeatedChars = 2 ibm-pwdPolicy = true	passwordMaxRepeatedChars = 3 ibm-pwdPolicy = true	passwordMaxRepeatedChars = 3 passwordMaxConsecutiveRepeatedChars = 0 ibm-pwdPolicy = true
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 the 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. For more information, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

Table 20 on page 115 displays examples of how a user's effective password policy is determined when individual, group, and global password policies are active.

Table 20. 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 20. 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 , 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 , 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 are in the critical access class and by default, only an LDAP administrator with the appropriate authority can query and read them. See Chapter 6, "Administrative group and roles," on page 91 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 10, "Using access control," on page 145 for more information.

Table 21. 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 ```

Table 21. Password policy operational attributes in user entries (continued)
Attribute and description
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: <pre>pwdAccountLockedTime: 20091021183747.488417Z</pre>
pwdExpirationWarned Specifies the Coordinated Universal Time in Zulu format of the first password expiration warning for this user. Example: <pre>pwdExpirationWarned: 20091021181746.852469Z</pre>
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: <pre>pwdFailureTime: 20091021181836.913647Z</pre>

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:
 	
<pre>pwdGraceUseTime: 20091013183626.310768Z
pwdGraceUseTime: 20091021155707.839414Z</pre>

Table 21. Password policy operational attributes in user entries (continued)

Attribute and description

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==
```

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 LDAPSRCH utility to retrieve operational attributes.

```
ldapsrch -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 LDAPSRCH utility to search for all entries when the password was last changed on May 2, 2010:

```
ldapsrch -D adminDn -w adminPw -s sub –b "c=us" "(!(pwdChangedTime>=20100502000000Z))" 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 LDAPSRCH utility to search for all entries that have an **pwdAccountLockedTime** attribute value:

```
ldapsrch -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 LDAPSRCH utility to search for all entries that have a **pwdReset** attribute value.

```
ldapsrch -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 LDAP client utilities send the **PasswordPolicy** server control on these requests to obtain additional warning and error information about password policy enforcement. For more information about the LDAP client utilities, see *z/VM: TCP/IP User's Guide*.

Table 22 on page 119 and Table 23 on page 120 contain summaries of the warnings and errors that are returned by the **PasswordPolicy** response control. See "PasswordPolicy" on page 348 for more information.

Table 22. **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 23. **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 6, "Administrative group and roles," on page 91 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 6, "Administrative group and roles," on page 91 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.
`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. For more information, see "Binding with SDBM using password policy" on page 62.

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 in 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 has been 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, it is recommended that peer-to-peer replication environments be used 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.

Enabling read-only replica password policy replication

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 6, "Administrative group and roles," on page 91 for more information about administrative role authority. The LDAPEXOP utility is provided to call these password policy extended operations. Table 24 on page 122 summarizes the extended operations including the LDAPEXOP operation value. For more information, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

Table 24. 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 the administrator, or the user's password is expired. See "Account status" on page 357 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 366 for more information.

See "Effective password policy examples" on page 115 and "Account status extended operation example" on page 127 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 6, "Administrative group and roles," on page 91 for more information about administrative role authority. This section shows examples of how the effective password policy is overridden for specific users.

The 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 LDAPMDFY utility to set the password expiration time to January 1, 2200 at midnight GMT.

```
ldapmdfy -D adminDn -w adminPw
dn: cn=user1,c=us
changetype: modify
replace: pwdChangedTime
pwdChangedTime: 22000101000000Z
```

The 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 LDAPMDFY utility to perform these modifications.

```
ldapmdfy -D adminDn -w adminPw
dn: cn=user2,c=us
changetype: modify
delete: pwdAccountLockedTime
-
delete: pwdFailureTime
```

The 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 LDAPMDFY 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 GMT.

```
ldapmdfy -D adminDn -w adminPw
dn: cn=user3,c=us
changetype: modify
replace: pwdChangedTime
pwdChangedTime: 20100601010000Z
-
replace: pwdExpirationWarned
-
replace: pwdGraceUseTime
```

The 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 LDAPMDFY utility to remove the **pwdReset** attribute.

```
ldapmdfy -D adminDn -w adminPw
dn: cn=user4,c=us
changetype: modify
delete: pwdReset
```

The LDAP administrator can force a user to change their password value by setting the **pwdReset** attribute value to true. This example uses the LDAPMDFY utility to set the **pwdReset** attribute value to true.

```
ldapmdfy -D adminDn -w adminPw
dn: cn=user5,c=us
changetype: modify
replace: pwdReset
pwdReset: true
```

The 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 LDAPMDFY utility to set the **ibm-pwdAccountLocked** attribute value to true.

```
ldapmdfy -D adminDn -w adminPw
dn: cn=user6,c=us
changetype: modify
replace: ibm-pwdAccountLocked
ibm-pwdAccountLocked: true
```

The 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 due to excessive password failures or an expired password.

```
ldapmdfy -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 LDAPMDFY 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

If the LDAP root administrator defined in the configuration file (**adminDN** configuration option) specifies an entry that resides in a CDBM or LDBM backend and the account becomes locked or expired under

z/VM LDAP password policy rules, the root administrator is unable to authenticate to the server. If this occurs, the LDAP server **UNLOCK** 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 command. In this example, *ldapsrv* is the name of the LDAP server virtual machine.

```
smsg ldapsrv 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 LDAPMDFY utility to activate the global password policy and to change its default values. For more information about the LDAPMDFY utility, see *z/VM: TCP/IP User's Guide*.

```
ldapmdfy -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 the LDAP root administrator or an administrator with the appropriate authority. See Chapter 6, "Administrative group and roles," on page 91 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, or 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. For more information about the LDAPADD client utility, see *z/VM: TCP/IP User's Guide*.

```
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 after 30 days (2592000) of last password modification.
- 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 needs to use this special password policy must be modified to set the **ibm-pwdGroupPolicyDN** operational attribute value. This example uses the LDAPMDFY 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.

```
ldapmdfy -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 LDAPMDFY utility to enable the evaluation of additional password policies in the LDAP server.

```
ldapmdfy -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 LDAPMDFY 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.

```
ldapmdfy -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 LDAPMDFY utility to enable the evaluation of additional password policies in the LDAP server.

```
ldapmdfy -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. For more information, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

The following 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 LDAPCHPW utility is provided and you must specify the current and new password values when modifying the **userPassword** attribute value. For more information, see *z/VM: TCP/IP User's Guide*.

The following example uses the LDAPCHPW utility to change the **userPassword** attribute value for user `cn=user1,c=us` from `secret` to `supersecret`.

For example:

```
ldapchpw -p port -D cn=user1,c=us -w secret -n supersecret
```

If changing passwords with a non-z/VM LDAP client, use the LDAPMDFY 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`.

```
ldapmdfy -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 8. CRAM-MD5 and DIGEST-MD5 authentication

The z/VM 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 (www.ietf.org/rfc/rfc2195.txt? number=2195). DIGEST-MD5 is defined in RFC 2831: Using Digest Authentication as a SASL Mechanism (www.ietf.org/rfc/rfc2831.txt?number=2831). 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/VM LDAP server do not require any additional products to be installed or configured.

The z/VM LDAP server DIGEST-MD5 bind mechanism supports the integrity and confidentiality options defined in RFC 2831: Using Digest Authentication as a SASL Mechanism (www.ietf.org/rfc/rfc2831.txt? number=2831). 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/VM 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/VM LDAP client and server.

2. Subsequent authentication is not supported.

Considerations for setting up an LDBM or CDBM backend for CRAM-MD5 and DIGEST-MD5 authentication

The following are considerations for setting up an LDBM or CDBM backend for CRAM-MD5 and DIGEST-MD5 authentication:

1. In order to use the CRAM-MD5 bind mechanism on the z/VM LDAP server, the LDBM or CDBM entry that you bind with should contain a **uid** attribute value. The **uid** attribute is always in the LDAP server schema. There are three ways to perform a CRAM-MD5 bind to the z/VM LDAP server:

   a. Only specifying the bindDN in the bind request in your client application. When using the z/VM LDAP client utilities, such as LDAPSRCH (**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 LDBM or CDBM entries. When using the z/VM LDAP client utilities, such as LDAPSRCH, 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 LDBM or CDBM entries. The bindDN specified in the bind request must map to the same distinguished name as the username. When using the z/VM LDAP client utilities, such as LDAPSRCH, this is done by specifying both the **-D** and the **-U** options.

For more information on the z/VM LDAP client utilities, see Using the LDAP Client Utilities in *z/VM: TCP/IP User's Guide*.

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/VM LDAP server, the LDBM or CDBM entries that you bind with must contain a **uid** attribute value. The **uid** attribute is always present in the server schema. There are two ways to perform a DIGEST-MD5 bind to the z/VM 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 LDBM or CDBM entries. When using the z/VM LDAP client utilities, such as LDAPSRCH, 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 LDBM or CDBM entries. The authorization DN that is specified must map to the same distinguished name as the username. When using the z/VM LDAP client utilities, such as LDAPSRCH, this is done by specifying both the **-D** and the **-U** options.

For more information on the z/VM LDAP client utilities, see Using the LDAP Client Utilities in *z/VM: TCP/IP User's Guide*.

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 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 recommended) or encrypted in either DES or AES. DES and AES encryption are recommended since they both encrypt the **userPassword** and provide clear text decryption. For additional information on AES and DES encryption, refer to Configuring for Encryption or Hashing in *z/VM: TCP/IP Planning and Customization*.

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 desired. See the digestRealm option in *z/VM: TCP/IP Planning and Customization*. 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 can not 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 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.
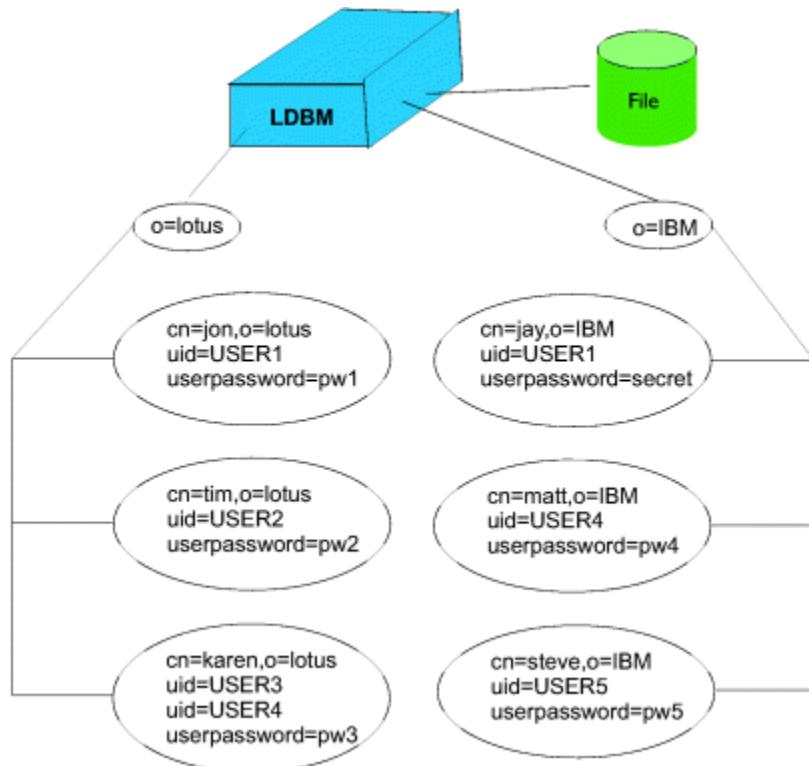

Figure 20. CRAM-MD5 and DIGEST-MD5 authentication example

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/VM LDAP client utilities, while the bindDN (CRAM-MD5) or authorization DN (DIGEST-MD5) is the **-D** option on the z/VM LDAP client utilities. See Using the LDAP Client Utilities in *z/VM: TCP/IP User's Guide* for more details on the LDAP client utilities. The following table assumes that native authentication is not turned on under the subtrees: o=lotus and o=IBM.

Table 25. 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

Table 25. Behavior of CRAM-MD5 and DIGEST-MD5 authentication in example (continued)			
Username	BindDN (CRAM-MD5) or authorization DN (DIGEST-MD5)	Password	Behavior
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 LDBM backend.

Chapter 9. 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 10, "Using access control," on page 145 for additional information on 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 in which the user has membership.

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 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 33. 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 will allow 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 LDAPSRCH in *z/VM: TCP/IP User's Guide* for additional information on LDAP search filters. The default is **"objectclass=*"**.

Note: As the format above suggests, the host name must not be present in the syntax. The remaining parameters are just like the normal LDAP URL syntax, defined in RFC 2255: The LDAP URL Format (www.ietf.org/rfc/rfc2255.txt?number=2255) (except there is no support for extensions in the URL). Each parameter field must be separated by a ?, even if no parameter is specified. Normally, a list of attributes to return would have been included between the *baseDN* and *searchScope*. 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.

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=holbrook
objectclass: groupOfURLs
cn: ldap_team_dynamic
memberURL: ldap:///o=holbrook??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=Holbrook,o=hal,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=Holbrook,o=hal,c=us??one
```

A subtree search for all the support staff in Holbrook:

```
ldap:///ou=Holbrook,o=hal,c=us??sub?title=*Support
```

A subtree search for all the Garcias or Nguyens whose first name begins with an "A":

```
ldap:///o=hal,c=us??sub?(&(|(sn=Garcia)(sn=Nguyen))(cn=A*))
```

A search filter that includes escaped percent signs, question marks and spaces in the base DN (o=deltawing infosystems) and filter ((&(percent=10%)(description=huh?))):

```
ldap:///o=deltawing%20infosystems,c=au??sub?(&(percent=10%25)(description=huh%3f))
```

Nested groups

A nested group is defined as a group that references other group entries, which can be static, dynamic, or nested groups. The **ibm-nestedGroup** object class uses the multi-valued attribute called **ibm-memberGroup** to indicate the DNs of the groups that are referenced by the nested group. 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 referenced within the nested group is ignored if it is not in the same backend as the nested group. The group hierarchy 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.

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=holbrook
objectclass: container
objectclass: ibm-nestedGroup
cn: ldap_team_nested
ibm-memberGroup: cn=ldap_team_static,o=holbrook
ibm-memberGroup: cn=ldap_team_dynamic,o=holbrook
ibm-memberGroup: cn=ldaptest_team_nested,o=holbrook
```

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 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 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 can not be specified on a search filter.

The **ibm-allGroups** and **ibm-allMembers** search and comparison operations are supported only on entries within the 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 at authentication time. The entries and attributes used to evaluate **ibm-allMembers** and **ibm-allGroups** have ACL restrictions, against which the bound DN has to 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 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 10, "Using access control," on page 145.

ACL restrictions on group gathering

If LDAP password policy is active, the list of the static, dynamic, and nested groups of which 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 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.

Using the limits from search limit groups

When a non-administrator user binds to the LDAP server 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 re-binds, 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 applied to the search are determined as follows:

1. The server first determines the server search limits. If the requestor 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 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, or nested group entry, delete the directory entry that represents the group. The **ldapdelete** command can be used to perform this delete operation.

This example deletes the static, dynamic, and nested group entries that were created in the above examples:

```
ldapdelete -h 127.0.0.1 -D "cn=admin" -w xxx -f deleteGrp.list
```

Where deleteGrp.list contains:

```
cn=nested_grp,o=Your Company
cn=group1,o=Your Company
cn=group2,o=Your Company
cn=group3,o=Your Company
cn=dynamic_team,o=Your Company
```

Examples of querying group membership
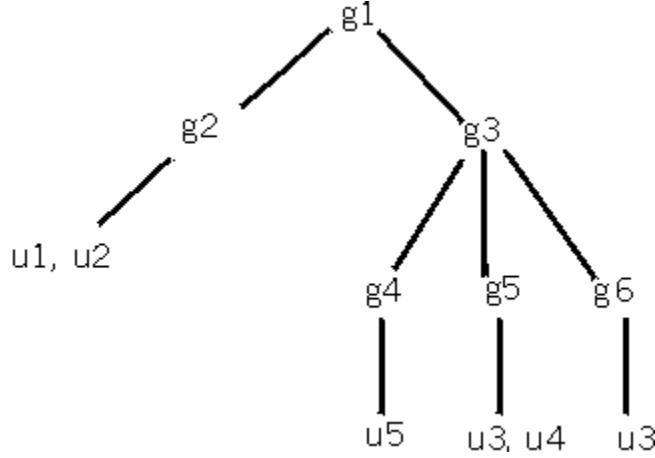
Figure 21. Group hierarchy and membership for the examples

The entries below are used in the following examples:

```
dn: o=hal
objectclass: organization
aclEntry: group:CN=ANYBODY:normal:rsc:system:rsc
aclPropagate: TRUE
o: ibm
```

```
dn: cn=g1,o=hal
objectclass: container
objectclass: ibm-nestedGroup
cn: g1
ibm-memberGroup: cn=g2,o=hal
ibm-memberGroup: cn=g3,o=hal
aclEntry: group:CN=ANYBODY:normal:rsc:system:rsc
```

```
dn: cn=g2,o=hal
objectclass: accessGroup
cn: g2
member: cn=u1,o=hal
member: cn=u2,o=hal
aclEntry: group:CN=ANYBODY:normal:rsc:system:rsc
aclEntry: access-id:cn=u1,o=hal:normal:rsc:system:rsc
aclEntry: access-id:cn=u2,o=hal:normal:rsc:system:rsc:at.member:deny:rsc
```

```
dn: cn=g3,o=hal
objectclass: container
objectclass: ibm-nestedGroup
cn: g3
ibm-memberGroup: cn=g4,o=hal
ibm-memberGroup: cn=g5,o=hal
ibm-memberGroup: cn=g6,o=hal
```

```
dn: cn=g4,o=hal
objectclass: accessGroup
cn: g4
aclEntry: group:CN=ANYBODY:normal:rsc:system:rsc
aclEntry: access-id:cn=u4,o=hal:normal:rsc:system:rsc:at.member:deny:c
member: cn=u5,o=hal
```

```
dn: cn=g5,o=hal
objectclass: container
objectclass: ibm-dynamicGroup
cn: g5
memberURL: ldap:///o=hal??sub?(|(cn=u3)(cn=u4))
aclEntry: group:cn=ANYBODY:normal:rsc:system:rsc
aclEntry: access-id:cn=u3,o=hal:normal:rsc:system:rsc:at.ibm-allMembers:deny:rs:
 at.ibm-allMembers:grant:c
```

```
dn: cn=g6,o=hal
objectclass: container
objectclass: ibm-dynamicGroup
cn: g6
memberURL: ldap:///o=hal??sub?(cn=*3)
```

```
dn: cn=u1,o=hal
objectclass: person
cn: u1
sn: user
userpassword: secret1
```

```
dn: cn=u2,o=hal
objectclass: person
cn: u2
sn: user
userpassword: secret2
```

```
dn: cn=u3,o=hal
objectclass: person
aclEntry: access-id:cn=u1,o=hal:normal:rsc:system:rsc:at.cn:deny:s
aclEntry: access-id:cn=u2,o=hal: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=hal
objectclass: person
aclentry: group:cn=ANYBODY:normal:rsc:system:rsc
aclentry: access-id:cn=u3,o=hal:normal:rsc:system:rsc:at.ibm-allGroups:deny:r
cn: u4
sn: user
userpassword: secret4
```

```
dn: cn=u5,o=hal
objectclass: person
cn: u5
sn: user
userpassword: secret5
```

```
dn: cn=u6,o=hal
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.

```
ldapsrch -L -D "cn=u6,o=hal" -w secret6 -b "cn=g4,o=hal" "objectclass=*" ibm-
allMembers

dn: cn=g4,o=hal
ibm-allmembers: cn=u5,o=hal
```

Access checking done for cn=u6,o=hal:

1. Read access to the **ibm-allMembers** attribute in cn=g4,o=hal.

2. Read access to the **member** attribute in cn=g4,o=hal.

Example 2: This example shows an **ibm-allMembers** attribute search on a dynamic group entry.

```
ldapsrch -L -D "cn=u6,o=hal" -w secret6 -b "cn=g5,o=hal" "objectclass=*" ibm-
allMembers
```

```
dn: cn=g5,o=hal
ibm-allmembers: cn=u3,o=hal
ibm-allmembers: cn=u4,o=hal
```

Access checking done for cn=u6,o=hal:

1. Read access to the **ibm-allMembers** attribute in cn=g5,o=hal.
2. Search access to the **cn** attribute in the returned entries, cn=u3,o=hal and cn=u4,o=hal, 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.

```
ldapsrch —L —D "cn=u6,o=hal" —w secret6 —b "cn=g3,o=hal" "objectclass=*" ibm-
allMembers

dn: cn=g3,o=hal
ibm-allmembers: cn=g3,o=hal
ibm-allmembers: cn=u3,o=hal
ibm-allmembers: cn=u4,o=hal
ibm-allmembers: cn=u5,o=hal
```

Access checking done for cn=u6,o=hal:

1. Read access to the **ibm-allMembers** attribute in cn=g3,o=hal.
2. Read access to the **member** attribute in cn=g4,o=hal.
3. Search access to the **cn** attribute in the returned entries, cn=u3,o=hal and cn=u4,o=hal, from the search filter specified in the **memberURL** attribute of cn=g5,o=hal.
4. Search access to the **cn** attribute in the returned entries, cn=u3,o=hal and cn=g3,o=hal, from the search filter specified in the **memberURL** attribute of cn=g6,o=hal.

   Note: Since cn=u3,o=hal has already been added as an **ibm-allMembers** attribute value, a duplicate value will not be added.

Note: ibm-memberGroup access rights do not matter.

Example 4: This example shows an **ibm-allMembers** attribute search on a dynamic group entry when the bound user is not granted read access to the **ibm-allMembers** attribute.

```
ldapsrch -L -D "cn=u3,o=hal" -w secret3 -b "cn=g5,o=hal" "objectclass=*" ibm-
allmembers

dn: cn=g5,o=hal
```

Access checking done for cn=u3,o=hal:

1. Read access to the **ibm-allMembers** attribute in cn=g5,o=hal has been denied. Therefore, no **ibm-allMembers** attribute values will be added.

Example 5: This example shows an **ibm-allMembers** attribute search on a static group entry when the bound user does not have read authority on the **member** attribute.

```
ldapsrch -L -D "cn=u2,o=hal" -w secret2 -b "cn=g2,o=hal" "objectclass=*" ibm-
allmembers

dn: cn=g2,o=hal
```

Access checking done for cn=u2,o=hal:

1. Read access to the **ibm-allMembers** attribute in cn=g2,o=hal.
2. Read access to the **member** attribute in cn=g2,o=hal has been denied. Therefore, the **member** attribute value will not be added as an **ibm-allMembers** attribute value.

Example 6: This example shows an **ibm-allMembers** attribute search on a dynamic group entry when the bound user does not have search authority in the entries that are to be returned for the attributes that are specified in the dynamic group filter.

```
ldapsrch -L -D "cn=u1,o=hal" -w secret1 -b "cn=g5,o=hal" "objectclass=*" ibm-
allmembers

dn: cn=g5,o=hal
ibm-allmembers: cn=u4,o=hal
```

Access checking done for cn=u1,o=hal:

1. Read access to the **ibm-allMembers** attribute in cn=g5,o=hal.
2. Search access to the **cn** attribute in the returned entries, cn=u3,o=hal and cn=u4,o=hal, from the search filter specified in the **memberURL** attribute. However, search access has been denied on the **cn** attribute of cn=u3,o=hal 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.

```
ldapcmpr -D "cn=u3,o=hal" -w secret3 "cn=g5,o=hal" "ibm-allmembers=cn=u3,o=hal"
ldap_compare: Compare true
```

Access checking done for cn=u3,o=hal:

1. Compare access to the **ibm-allMembers** attribute in cn=g5,o=hal.
2. Search access to the **cn** attribute on the returned entries, cn=u3,o=hal and cn=u4,o=hal, 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.

```
ldapsrch -L -D "cn=u6,o=hal" -w secret6 -b "cn=u4,o=hal" "objectclass=*" ibm-
allGroups

dn: cn=u4,o=hal
ibm-allgroups: cn=g5,o=hal
ibm-allgroups: cn=g3,o=hal
ibm-allgroups: cn=g1,o=hal
```

Access checking done for cn=u6,o=hal:

1. Read access to the **ibm-allGroups** attribute in cn=u4,o=hal.
2. Search access on the **cn** attribute in cn=u4,o=hal from the search filter specified in the **memberURL** attribute in cn=g5,o=hal.

Since cn=g3,o=hal has cn=g5,o=hal as an **ibm-memberGroup** attribute value, cn=g3,o=hal is added as an **ibm-allGroups** attribute also. cn=g1,o=hal has cn=g3,o=hal as an **ibm-memberGroup** value, therefore cn=g1,o=hal 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.

```
ldapsrch -L -D "cn=u1,o=hal" -w secret1 -b "cn=u2,o=hal" "objectclass=*" ibm-
allGroups

dn: cn=u2,o=hal
ibm-allgroups: cn=g2,o=hal
ibm-allgroups: cn=g1,o=hal
```

Access checking done for cn=u1,o=hal:

1. Read access to the **ibm-allGroups** attribute in cn=u2,o=hal.
2. Read access to the **member** attribute in cn=g2,o=hal.

Since `cn=g1,o=hal` has an **ibm-memberGroup** attribute value of `cn=g2,o=hal`, `cn=g1,o=hal` 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.

```
ldapsrch -L -D "cn=u2,o=hal" -w secret2 -b "cn=u2,o=hal" "objectclass=*" ibm-
allGroups

dn: cn=u2,o=hal
```

Access checking done for `cn=u2,o=hal`:

1. Read access to the **ibm-allGroups** attribute in `cn=u2,o=hal`.
2. Read access to the **member** attribute of `cn=g2,o=hal` is denied. Therefore, `cn=g2,o=hal` 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.

```
ldapsrch -L -D "cn=u3,o=hal" -w secret3 -b "cn=u4,o=hal" "objectclass=*" ibm-
allGroups

dn: cn=u4,o=hal
```

Access checking done for `cn=u3,o=hal`:

1. Read access to the **ibm-allGroups** attribute in `cn=u4,o=hal` 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.

```
ldapcmpr -D "cn=u2,o=hal" -w secret2 "cn=u3,o=hal" "ibm-allGroups=cn=g1,o=hal"
```

```
ldap_compare: Compare true
```

Access checking done for `cn=u2,o=hal`:

1. Compare access to the **ibm-allGroups** attribute in `cn=u3,o=hal`.
2. Search access to the **cn** attribute of `cn=u3,o=hal` is granted from the search filter specified in the **memberURL** attribute in `cn=g5,o=hal`.

Since `cn=g3,o=hal` has `cn=g5,o=hal` as an **ibm-memberGroup** attribute value, `cn=g3,o=hal` is added as an **ibm-allGroups** attribute as well. `cn=g1,o=group` has `cn=g3,o=hal` as an **ibm-memberGroup** value, therefore `cn=g1,o=group` is also added as an **ibm-allGroups** attribute value. Therefore, the compare operation will return an **LDAP_COMPARE_TRUE** to the client application.

Chapter 10. Using access control

Access control of information in the LDAP server is specified by setting up Access Control Lists (ACLs). LDBM, GDBM, or CDBM ACLs provide a means to protect information stored in an LDAP directory. Administrators use ACLs to restrict access to different portions of the directory, or specific directory entries. When using the LDBM, GDBM, or CDBM backend, ACLs are created and managed using the `ldap_add` and `ldap_modify` APIs.

ACLs are represented by a set of attributes which appear to be a part of the entry. The attributes associated with access control, such as `entryOwner`, `ownerPropagate`, `aclEntry`, and `aclPropagate`, are unusual in that they are logically associated with each entry, but can have values which depend upon other entries higher in the directory hierarchy. Depending upon how they are established, these attribute values can be explicit to an entry, or inherited from an ancestor entry.

Use of LDAP's SDBM backend allows a user to be authenticated to the directory namespace using the RACF ID and password. The RACF identity becomes associated with the user's RACF-style distinguished name that was used on the LDAP bind operation. It is then possible to set up ACLs for entries managed by the LDBM, GDBM, or CDBM backend using RACF-style user and group DNs. This controls access to LDBM, GDBM, or CDBM database directory entries 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 which describe who is allowed to access information within that entry. Table 26 on page 145 shows the set of attributes which 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 26. ACL and entry owner attributes

Attribute	Information
ACL attributes	
`aclEntry`	This is a multi-valued attribute that contains the names and permissions associated with those names that have access to information in the directory entry (or the entry along with the subtree of information below the entry, depending on the setting of the `aclPropagate` attribute).
`aclPropagate`	This is a single-valued boolean attribute which indicates whether the `aclEntry` information applies only to the directory entry it is associated with or to the entire subtree of information including and below the directory entry it is associated with. Note that propagation does not apply to entries that have an explicit **aclEntry** defined for the entry and that propagation stops at the next propagating ACL (`aclPropagate=TRUE`) that is encountered in the directory subtree.
`aclSource`	This is a single-valued attribute that is managed by the LDAP server and cannot be changed by the LDAPMDFY command. This attribute, accessible for any directory entry, indicates the distinguished name of the entry that holds the ACL that applies to the entry. This attribute is useful in determining which propagating ACL is used to control access to information in the directory entry.
Entry owner attributes	

Table 26. ACL and entry owner attributes (continued)

Attribute	Information
entryOwner	This is a multi-valued attribute that contains the distinguished names of users or groups that are considered owners of the directory entry (or the entry along with the subtree of information below the entry, depending on the setting of the ownerPropagate attribute).
ownerPropagate	This is a single-valued boolean attribute which indicates whether the entryOwner information applies only to the directory entry it is associated with or to the entire subtree of information including and below the directory entry it is associated with. Note that propagation does not apply to entries that have an explicit entryOwner defined for the entry and that propagation stops at the next propagating entryOwner (ownerPropagate=TRUE) that is encountered in the directory subtree.
ownerSource	This is a single-valued attribute that is managed by the LDAP server and cannot be changed by the LDAPMDFY command. This attribute indicates the distinguished name of the entry that holds the entryOwner that applies to the entry. This attribute is useful in determining which propagating entryOwner is used to control access to information in the directory entry.

aclEntry attribute

aclEntry is a multi-valued attribute which 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*

  where,

  aclEntry_value :- [**access-id:**|**group**:|**role**:]*subject_DN*[*granted_rights*]

  or,

  aclEntry_value :- **aclFilter**:*filter*:*operation*[*granted_rights*]
- *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 151 for more information.
- *operation* :- **union** | **replace** | **intersect**. See "ACL filters" on page 151 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 which represents the object (entry) to which privileges are being granted. The DN ends when the first granted rights keyword is detected.

The *granted_rights* is specified as follows where *object_rights_list* is one or more elements of the set [a|d], and *attr_rights_list* is one or more elements of the set [r|w|s|c].

See "ACL filters" on page 151 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 will be merged into a single specification. For example:

```
group:cn=Anybody:normal:rs:system:rsc:normal:c:normal:deny:w
```

will result 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 three 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.

Access control groups can be either static, dynamic, or nested groups. The following object classes are evaluated as group entries for the LDBM and CDBM backends: **ibm-staticGroup, groupOfNames, groupOfUniqueNames, accessRole, accessGroup, ibm-dynamicGroup, groupOfUrls,** and **ibm-nestedGroup**. For additional information on static, dynamic, and nested groups, see Chapter 9, "Static, dynamic, and nested groups," on page 133.

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 is 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 refer to large numbers of subject DNs which, at bind time, share a common characteristic in relation to either the operation being performed or the target object on which the operation is being performed. Currently, three pseudo DNs are defined:

```
group:cn=anybody
group:cn=authenticated
access-id:cn=this
```

The `group:cn=anybody` refers to all subjects, including those that are unauthenticated (considered anonymous users). All users belong to this group automatically. The `group:cn=authenticated` refers to any DN which has been authenticated to the directory. The method of authentication is not considered.

The `access-id:cn=this` refers to the bind DN which matches the target object's DN on which the operation is performed.

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 151 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=HAL, c=US`.

```
access-id:cn=personA, ou=deptXYZ, o=HAL, c=US
```

In this example, the DN type is `group` and the DN itself is `cn=deptXYZRegs, o=HAL, c=US`.

```
group:cn=deptXYZRegs, o=HAL, c=US
```

This is an example of how to use a RACF identity established with SDBM in an ACL.

```
access-id:racfid=YourID,profileType=user
group:racfid=YourGroup,profileType=group
```

Attribute access classes

Attributes requiring similar permission for access are grouped together in classes. Attributes are assigned to an attribute access class within the schema definitions. The `IBMAttributeTypes` attribute in the LDAP server schema entry holds the attribute type's access class. The three attribute access classes are:

- `normal`
- `sensitive`
- `critical`

Each of these attribute access classes is discrete. If a user has write permission to `sensitive` attributes, then the user does not automatically have write permission to `normal` attributes. This permission must be explicitly defined.

The default attribute access class for an attribute is `normal`. By default, all users have read access to `normal` attributes. There are two additional attribute access classes used internally by LDAP for system attributes. These attribute access classes are `restricted` and `system`. You can specify these access classes when granting permissions in ACLs.

For example, a person's name would typically be defined in the `normal` class. Perhaps a social security number would be considered `sensitive`, and any password information for the user would be considered `critical`. Following are some example definitions excerpted from 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. In order to update access control information, you must have permissions to read and write these attributes. The **system** attributes include aclSource and ownerSource and other attributes for which the server controls the values. In order to update access control information, you must have permission to read and write these attributes. If the **system** keyword is not specified in an **aclEntry** attribute value, the system access will be set to 'system:rsc'.

Access permissions

Following is the set of access permissions.

Table 27. Permissions which apply to an entire entry

Permission	Meaning
Add	Add an entry below this entry
Delete	Delete this entry

Table 28. Permissions which apply to attribute access classes

Permission	Meaning
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 155 for information on 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

Each entry with an explicit ACL has associated with it an `aclPropagate` attribute. By default, the entry's explicit ACL is inherited down the hierarchy tree, and its **aclPropagate** attribute is set to TRUE. If set to FALSE, the explicit ACL for the entry becomes an override, pertaining only to the particular entry. The `aclPropagate` syntax is Boolean. See "Propagating ACLs" on page 164 for more information.

aclSource attribute

Each entry has an associated `aclSource`. This reflects the DN with which the ACL is associated. This attribute is kept and managed by the server, but may be retrieved for administrative purposes. This attribute cannot be set, only retrieved.

The derivation of `aclSource` is further explained in "Propagating ACLs" on page 164.

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.

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 is 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.

Note: Unlike the **aclEntry** attribute, the **entryOwner** attribute cannot reduce permissions for the LDAP administrator.

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 6, "Administrative group and roles," on page 91 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.

Syntax

Following is the `entryOwner` attribute value syntax:

- **entryOwner:** *entryOwner_value*

  where,

  entryOwner_value :- [**access-id:**|**group**:|**role**:]*subject_DN*

  or,

  entryOwner_value :- **ownerFilter:***filter*:**grant**|**deny**

- *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 151 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.

ownerPropagate attribute

Owner propagation works exactly the same as ACL propagation. By default, owners are inherited down the hierarchy tree, and their owner propagate attribute is set to TRUE. If set to FALSE, the owner becomes an override, pertaining only to the particular entry. The `ownerPropagate` syntax is boolean.

ownerSource attribute

Each entry also has an associated `ownerSource`. This reflects the DN with which the owner values are associated. This attribute is kept and managed by the server, but can be retrieved for administrative purposes. This attribute cannot be set, only retrieved.

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 syntax of specifying a filter in an **aclEntry** attribute is:

aclFilter:*filter*:*operation*[*granted_rights*]

See "aclEntry attribute" on page 146 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 (www.ietf.org/rfc/rfc2254.txt?number=2254)) compliant LDAP search filter 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 (www.ietf.org/rfc/rfc2373.txt?number=2373)). 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 normalized to their fully expanded syntax. 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, normalization shifts the wildcard to the final group of the address. Following are some examples of normalization:

Table 29. **ibm-filterIP** *normalization examples*

Not normalized address	Normalized 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**, and **DIGEST-MD5**. This attribute 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 the administrator DN.

Initializing ACLs with LDBM

The LDBM backend adds an ACL to each suffix entry if no **aclEntry** value is specified during the add of this entry. 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

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 sever 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 change log 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 6, "Administrative group and roles," on page 91 for more information about administrative role authority.

The **aclEntry** and **entryOwner** attributes can be entirely deleted, in which case the default ACL is used. See "Default ACLs with LDBM" on page 153 for more information.

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`
- `cn=configuration`
- `cn=Replication,cn=configuration`
- `cn=Log Management,cn=configuration`
- `cn=Replication,cn=Log Management,cn=configuration`
- `cn=admingroup,cn=configuration`
- `cn=safadmingroup,cn=configuration`

The `cn=ibmpolicies` suffix entry is created with the same initial ACL as an 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 6, "Administrative group and roles," on page 91 for more information about administrative role authority.

Initializing ACLs with schema entry

When the LDAP sever 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 an LDBM suffix that allows read access to anybody. Therefore, only the LDAP root or schema administrator can update the schema. The **aclEntry** and **entryOwner** values can be modified.

Access determination

The same bound user may 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

For using the **GetEffectiveACL** extended operation in the LDAPEXOP utility, see "Querying effective permissions" on page 161.

The z/VM 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
- Whether 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 the 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 the LDAP administrator with the appropriate authority (see Chapter 6, "Administrative group and roles," on page 91 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 has not yet been 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.
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.
4. If a base **aclEntry** value matches the alternate DN(s) or filter in an **aclFilter** component matches the alternate DN(s) and additional access information, then:

   - See Step b.
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.
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.
7. If a base **aclEntry** value matches the group DN(s) that the bind or alternate DNs belongs to or filter in an **aclFilter** component match the group DN(s) and additional access information, then:

   - See Step b.
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.
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.
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 algorithm applies permissions and ends as follows:
  - The union of 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.

Note: In each of these steps, permissions for matching filters can be set if base permissions have already been 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=HAL,c=US:object:ad:normal:rwsc:sensitive:rwsc:critical:rsc
```

In this example, the user corresponding to `cn=personA, ou=deptXYZ, o=HAL, 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(s).

Example 5:

```
aclEntry: cn=Karen,dc=yourcompany,dc=com:normal:rwsc:sensitive:rsc:at.userpassword:w:
 critical:deny:rwsc
```

In this example, cn=Karen,dc=yourcompany,dc=com is granted read, search, and compare to **normal** and **sensitive** attributes, and write to **normal** attributes and the **userpassword** attribute. All access to **critical** attributes (except for write in **userpassword**) is turned off.

Example 6:

```
aclEntry: group:cn=group1,dc=yourcompany,dc=com:normal:rwsc
aclEntry: group:cn=group2,dc=yourcompany,dc=com:sensitive:rwsc:at.cn:deny:w
```

In this example, a member of group1 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 AM

to 5:00 PM. 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 AM to 5:00 PM, 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=HAL: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=HAL` 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=HAL` 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=HAL`, therefore, only that base `access-id` ACL is granted, and no group ACLs are granted. `cn=Joe,dc=yourcompany,dc=com,o=HAL`'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=HAL`. 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=HAL: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=HAL` (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=HAL` is granted read permission on **normal** attributes. This is because `cn=Mary,dc=yourcompany,dc=com,o=HAL`'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 together 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 her base effective permission.

3. The union filters that match for `cn=Mary, o=Your Company` are then joined together 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 together 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 her 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

```
ldapsrch -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. For more information, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

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 descendents 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 will propagate down to all descendants. If set to FALSE, the ACL is not propagated; it instead becomes an override ACL. The ACL is not propagated down through the hierarchy, but instead applies only to the one particular entry that it is associated with within the hierarchy. If unspecified, `aclPropagate` is set to TRUE.

An entry without an explicit ACL receives its ACL from the nearest propagating ancestor ACL. 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 down 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=HAL, c=US` :

```
aclPropagate:  TRUE
aclEntry: group:cn=deptXYZRegs, o=HAL, c=US:normal:rcs:sensitive:rsc
aclEntry: access-id:cn=personA, ou=deptXYZ, o=HAL, c=US:object:ad:normal:rwsc:sensitive:rwsc:critical:rsc
aclEntry: group:cn=Anybody:normal:rsc
aclSource: ou=deptXYZ, o=HAL, c=US
```

In the absence of an explicit ACL for entry `cn=personA, ou=deptXYZ, o=HAL, c=US`, the following is the implicit, propagated ACL for the entry:

```
aclPropagate: TRUE
aclEntry: group:cn=deptXYZRegs, o=HAL, c=US:normal:rcs:sensitive:rsc
aclEntry: access-id:cn=personA, ou=deptXYZ, o=HAL, c=US:object:ad:normal:rwsc:sensitive:rwsc:critical:rsc
aclEntry: group:cn=Anybody:normal:rsc
aclSource:ou=deptXYZ, o=HAL, c=US
```

In this example, a propagating ACL has been set on `ou=deptXYZ, o=HAL, c=US`. No ACL has been set on the descendant `cn=personA, ou=deptXYZ, o=HAL, c=US`. Therefore, the descendant inherits its ACL value from the nearest ancestor with a propagating ACL. This happens to be `ou=deptXYZ, o=HAL, 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=HAL, c=US`.

Examples of overrides

Following is an explicit ACL for entry `o=HAL, c=US`:

```
aclPropagate:  TRUE
aclEntry:    group:cn=IBMRegs, o=HAL, c=US:normal:rcs:sensitive:rsc
aclEntry:    group:cn=Anybody:normal:rsc
aclSource:   o=HAL, c=US
```

Following is an explicit ACL for entry `ou=deptXYZ, o=HAL, c=US`:

```
aclPropagate: FALSE
aclEntry: group:cn=deptXYZRegs, o=HAL, c=US:normal:rcs:sensitive:rsc
aclEntry: access-id:cn=personA, ou=deptXYZ, o=HAL, c=US:object:ad:normal:rwsc:sensitive:rwsc:critical:rsc
aclEntry: group:cn=Anybody:normal:rsc
aclSource: ou=deptXYZ, o=HAL, 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=HAL, c=US`:

```
aclPropagate:  TRUE
aclEntry:  group:cn=IBMRegs, o=HAL, c=US:normal:rcs:sensitive:rsc
aclEntry:  group:cn=Anybody:normal:rsc
aclSource:  o=HAL, c=US
```

In this example, a propagating ACL has been set on o=HAL, c=US. An override ACL has been set (`aclPropagate` is FALSE) on the descendant ou=deptXYZ, o=HAL, c=US. Therefore, the ACL set at ou=deptXYZ, o=HAL, c=US pertains only to that particular entry.

The descendant cn=personA, ou=deptXYZ, o=HAL, c=US inherits its ACL value from the nearest ancestor with a propagating ACL (which is o=HAL, c=US as reflected in the `aclSource`). The ACL on ou=deptXYZ, o=HAL, 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=HAL, c=US:

```
aclPropagate:  TRUE
aclEntry:  group:cn=deptXYZRegs, o=HAL, c=US:normal:rcs:sensitive:rsc
aclEntry:  access-id:cn=personA, ou=deptXYZ, o=HAL, c=US:object:ad:normal:rwsc:sensitive:rwsc:critical:rsc
aclEntry:  group:cn=Anybody:normal:rsc:system:rsc
aclSource:  ou=deptXYZ, o=HAL, c=US
ownerPropagate:  TRUE
entryOwner:  access-id:cn=deptXYZMgr, ou=deptXYZ, o=HAL, c=US
ownerSource:  ou=deptXYZ, o=HAL, c=US
```

This is an inherited ACL and an inherited owner. Both owner properties and ACL properties are inherited from entry ou=deptXYZ, o=HAL, c=US. In this example, members of group cn=deptXYZRegs, o=HAL, 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 and CDBM 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 9, "Static, dynamic, and nested groups," on page 133 for more information on 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.

- 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, or CDBM 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 username. If a DN is specified, there must be an entry in LDAP with that DN. If a username is specified, there must be an entry in LDAP that contains the username as a **uid** attribute value. If both a DN and a username are specified, there must be an entry in LDAP with that DN and the username 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 or CDBM backend, or in a client operation plug-in extension. There are no alternate DNs. See Chapter 8, "CRAM-MD5 and DIGEST-MD5 authentication," on page 129 for more information.

- For a DIGEST-MD5 bind, the bind request must specify a username and may optionally contain an authorization DN. If only a username is specified, there must be an entry in LDAP that contains the username as a **uid** attribute value. If both a username and an authorization DN are specified, there must be an entry in LDAP with the authorization DN as its DN and the username 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 or CDBM backend or in a client operation plug-in extension. See Chapter 8, "CRAM-MD5 and DIGEST-MD5 authentication," on page 129 for more information.

- For a certificate (EXTERNAL) bind, the bind DN is normally the subject DN from the certificate specified in the bind request. There can not be an entry in an LDBM or CDBM backend or in a client operation plug-in extension corresponding to this DN.

After the bind DN is 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, this step is skipped.

- Each LDBM or CDBM 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 have 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 subsequently created with the same DN, that user or group will be granted the privileges of the former user or group.

Retrieving ACL information from the server

In order to retrieve all of the ACL information in a namespace, use the LDAPSRCH command, as shown in the following example:

```
ldapsrch -h 127.0.0.1 -D "cn=admin, dc=Your Company,dc=com" -w xxxxxx
 -b "dc=Your Company,dc=com" "(objectclass=*)" aclEntry aclPropagate aclSource
 entryOwner ownerPropagate ownerSource
```

```
dn: dc=Your Company, dc=com
aclPropagate: TRUE
aclEntry: CN=ADMIN:normal:rwsc:sensitive:rwsc:critical:rwsc:object:ad
aclEntry: CN=ANYBODY:normal:rsc:system:rsc
aclSource: dc=Your Company, dc=com
ownerPropagate: TRUE
entryOwner: CN=ADMIN
ownerSource: default
```

This command performs a subtree search starting at the root of the tree (assuming the root of the tree is `"dc=Your Company,c=com"`) and returns the six ACL attributes for each entry in the tree. It is necessary to specifically request the six ACL attributes because they are considered as "operational" and, therefore, can only be returned on a search if requested. (See IETF RFC 2251, RFC 2251: Lightweight Directory Access Protocol (v3) (www.ietf.org/rfc/rfc2251.txt?number=2251).)

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 requestor's group search limits, then some entries may not be returned. For more information, see the **sizeLimit** configuration option in Configuring the LDAP Server in *z/VM: TCP/IP Planning and Customization*.

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 `search` command, where *searchbase* is the starting point for the search.

Creating and managing access controls

To create and update ACLs in LDBM, GDBM, CDBM, or the schema entry, use a tool implementing `ldap_modify` APIs, such as LDAPMDFY (the **ldapmodify** utility) allows creation, modification, and deletion of any set of attributes that are associated with an entry in the directory. Since access control

information is maintained as a set of additional attributes within an entry, LDAPMDFY is a natural choice for administering access control information in LDBM, GDBM, CDBM, or the schema entry.

For details on using the utilities, such as LDAPMDFY, see *z/VM: TCP/IP User's Guide*.

Creating an ACL

In order to create an ACL, the `aclEntry` and `aclPropagate` attributes must be added to the information stored for an entry. The `aclEntry` and `aclPropagate` attributes are added to an entry by either specifying them as part of the entry information when the entry is added to the directory or by modifying the entry after it exists to contain the `aclEntry` and `aclPropagate` information.

It is possible to create an ACL without specifying the `aclPropagate` attribute. In this case, the `aclPropagate` attribute is assumed to have a value of TRUE and is added into the directory entry automatically, along with the `aclEntry` information.

Since LDAPMDFY is very powerful, all the possible ways of adding the `aclEntry` and `aclPropagate` information cannot be shown here. The examples shown here describe the more common uses of LDAPMDFY ato add ACL information.

Figure 22 on page 168 shows how to add a propagating ACL with three `aclEntry` values to an existing entry replacing any current **aclEntry** value.

```
ldapmdfy -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 22. Example of adding propagating ACL to existing entry in directory

The ACL added in Figure 22 on page 168 is created as a propagating ACL since the `aclPropagate` attribute is not specified and so assumed to be TRUE. This means that the ACL will apply to all entries below `cn=tim, o=Your Company` that do not already have an ACL associated with them. Note that the first and last `aclEntry` values span two lines in the `newAcl.ldif` file. In order to do this, the first character on the continued line must be a space character, as shown in the example.

While it is not required that the administrator update all ACL information, the examples in this section all use the administrator when updating ACLs. Further, the use of `-h 127.0.0.1` implies that the LDAPMDFY commands are performed from the same system on which the LDAP server is running and that the LDAP server is listening on TCP/IP port 389. For more details on the `-h`, `-p`, `-D`, and `-w` command-line options, refer to the LDAPMDFY command description in *z/VM: TCP/IP User's Guide*. The ACL attributes can be updated from any LDAP client as long as the user performing the updates has the proper authorization to update the ACL information.

The ACL attributes are defined to be in a special access class called `restricted`. Therefore, in order to allow someone other than the LDAP administrator to update the ACL attributes, they must either be the entry owner or have the proper authorization to `restricted` attributes. Figure 23 on page 169 shows an example of adding an ACL so that `cn=jeanne, o=Your Company` can update the ACL information:

```
ldapmdfy -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 23. Example of adding propagating ACL to existing entry in the directory

The ACL added in Figure 23 on page 169 allows `cn=jeanne, o=Your Company` to update the ACL information for this entry. In addition, since the ACL is a propagating ACL, this allows `cn=jeanne, o=Your Company` to create new ACL information against any entry that is controlled by this ACL. Care must be taken here, however, since it is possible for `cn=jeanne, o=Your Company` to set up an ACL which then does not allow `cn=jeanne, o=Your Company` update capability on the ACL information. If this occurs, a user with sufficient authority (the administrator, for example) must be used in order to reset/change the ACL information.

Figure 24 on page 169 shows an example of adding a non-propagating ACL. A non-propagating ACL applies only to the entry to which it is attached and not to the subtree of information that might be stored below the entry in the directory.

```
ldapmdfy -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 24. Example of setting up a non-propagating ACL

Setting up a non-propagating ACL is similar to setting up a propagating ACL. The difference is that the `aclPropagate` attribute value is set to FALSE.

Modifying an ACL

Once an ACL exists for an entry in the directory, it may have to be updated. To do this, the LDAPMDFY command is used. The examples in this section use the LDAPMDFY command, however, any LDAP client application issuing LDAP modify operations to the LDAP server may be used. Therefore, modifications to ACL information need not be performed from the same system on which the LDAP server is running.

Modifications to ACLs can be of a number of different types. The most common modifications are to:

- Add an additional `aclEntry` value to the ACL to allow another person or group access to the entry
- Change an ACL from propagating to non-propagating (not permitted for the GDBM change log suffix, `cn=changelog`)

- Remove an `aclEntry` value which exists in the ACL to disallow another person or group access to the entry that they had before.

Figure 25 on page 170, Figure 26 on page 170, and Figure 27 on page 170 show examples of these modifications, respectively.

"Access determination" on page 155 shows how an additional `aclEntry` value is added to existing ACL information.

```
ldapmdfy -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 25. Example of adding an aclEntry attribute value

In Figure 25 on page 170, 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 26 on page 170 shows how to modify an existing ACL to be non-propagating instead of propagating.

```
ldapmdfy -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 26. Example of modifying aclPropagate attribute

In Figure 26 on page 170, the existing ACL against cn=tim, o=Your Company is modified to be a non-propagating ACL instead of a propagating ACL. This means that the ACL will no longer apply to entries below cn=tim, o=Your Company in the directory tree. Instead, the first propagating ACL that is found in an entry above cn=tim, o=Your Company will be applied to the entries below cn=tim, o=Your Company. If no propagating ACL is found in the entries above cn=tim, o=Your Company, then the default ACL is used.

Figure 27 on page 170 shows how to remove an `aclEntry` value from existing ACL information:

```
ldapmdfy -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 27. Example of removing a single aclEntry attribute value

In Figure 27 on page 170, 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 needs to 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 LDAPMDFY command to delete the entire attribute (all values) from the entry.

Figure 28 on page 171 shows an example of deleting an ACL from an entry.

```
ldapmdfy -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 28. Example of deleting an ACL from an entry

In Figure 28 on page 171, the existing ACL against `cn=jeff, cn=tim, o=Your Company` is removed. This means that the ACL will no longer apply to the entry. Instead, the first propagating ACL that is found in an entry above `cn=jeff, cn=tim, o=Your Company` will be applied to `cn=jeff, cn=tim, o=Your Company`. If no propagating ACL is found in the entries above `cn=jeff, cn=tim, o=Your Company`, then the default ACL is used.

Creating an owner for an entry

In addition to the access control list control of directory entries, each entry can have assigned to it an entry owner or set of entry owners. As an entry owner, full access is allowed to the entry. Entry owners are granted add and delete permission, as well as read, write, search, and compare for all attribute classes. Entry owners can add and modify ACL information on the entries for which they are specified as the owner.

Entry owners are listed in the `entryOwner` attribute. Just like `aclEntry` information, `entryOwner` information can be propagating or non-propagating based on the setting of the `ownerPropagate` attribute. Like the `aclSource` attribute for `aclEntry` information, the `ownerSource` attribute lists the distinguished name of the entry that contains the `entryOwner` attribute which applies to the entry. The **ownerSource** attribute is set by the server and cannot be directly set when modifying the ACLs.

In order to create an entry owner, the `entryOwner` and `ownerPropagate` attributes must be added to the information stored for an entry. The `entryOwner` and `ownerPropagate` attributes are added to an entry by either specifying them as part of the entry information when the entry is added to the directory or by modifying the entry after it exists to contain the `entryOwner` and `ownerPropagate` information.

It is possible to create an entry owner without specifying the `ownerPropagate` attribute. In this case, the `ownerPropagate` attribute is assumed to have a value of TRUE and is added into the directory entry automatically.

Since the LDAPMDFY command is very powerful, all the possible ways of adding the `entryOwner` and `ownerPropagate` information cannot be shown here. The examples shown here describe the more common uses of the LDAPMDFY command to add entry owner information.

Figure 29 on page 172 shows how to add a propagating entry owner with two `entryOwner` values to an existing entry.

```
ldapmdfy -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 29. Example of adding a propagating set of entry owners to existing entry in the directory

The entry owners added in Figure 29 on page 172 are created as a propagating set of entry owners since the `ownerPropagate` attribute is not specified and so assumed to be **TRUE**. This means that the entry owners will apply to all entries below `cn=tim, o=Your Company` that do not already have an entry owner associated with them.

While it is not required that the LDAP administrator update all entry owner information, the examples in this section all use the administrator as the entry owner updating ACLs. Further, the use of `-h 127.0.0.1` implies that the LDAPMDFY commands are performed from the same system on which the LDAP server is running and that the LDAP server is listening on TCP/IP port 389. For more details on the `-h`, `-p`, `-D`, and `-w` command-line options, refer to the LDAPMDFY command description in *z/VM: TCP/IP User's Guide*. The entry owner attributes can be updated from any LDAP client as long as the user performing the update has the proper authorization to update the entry owner information.

The entry owner attributes, like the ACL attributes, are defined to be in a special access class called `restricted`. Therefore, in order to allow someone other than the LDAP administrator to update the entry owner attributes, they must either be the entry owner or have the proper authorization to `restricted` attributes. See Figure 23 on page 169 for an example of allowing users other than the LDAP administrator the ability to update entry owner information.

Figure 30 on page 172 shows an example of adding a non-propagating entry owner. A non-propagating entry owner applies only to the entry to which it is attached and not to the subtree of information that might be stored below the entry in the directory.

```
ldapmdfy -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 30. Example of setting up a non-propagating entry owner

Setting up a non-propagating entry owner is similar to setting up a propagating entry owner. The difference is that the `ownerPropagate` attribute value is set to FALSE.

Modifying an owner for an entry

Once an entry owner exists for an entry in the directory, it may have to be updated. To do this, the LDAPMDFY command is used. The examples in this section use the LDAPMDFY command, however, any LDAP client application issuing LDAP modify operations to the LDAP server may be used. Therefore, modifications to entry owner information need not be performed from the same system on which the LDAP server is running.

Modifications to entry owners can be of a number of different types. The most common modifications are to:

- Add an additional `entryOwner` value to the set of entry owners to allow another person or group to control the entry
- Change an entry owner from propagating to non-propagating (not permitted for the GDBM change log suffix, `cn=changelog`)
- Remove an `entryOwner` value which exists in the entry owner set to disallow another person or group access to control the entry that they had control over before.

Figure 31 on page 173, Figure 32 on page 173, and Figure 33 on page 174 show examples of these modifications, respectively.

Figure 31 on page 173 shows how an additional `entryOwner` value is added to existing entry owner information.

```
ldapmdfy -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 31. Example of adding an entryOwner attribute value

In Figure 31 on page 173, `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 32 on page 173 shows how to modify an existing entry owner to be non-propagating instead of propagating.

```
ldapmdfy -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 32. Example of modifying the ownerPropagate attribute

In Figure 32 on page 173, the existing entry owner set for `cn=tim, o=Your Company` is modified to be non-propagating instead of propagating. This means that the entry owner will no longer apply to entries below `cn=tim, o=Your Company` in the directory tree. Instead, the first propagating entry owner set that is found in an entry above `cn=tim, o=Your Company` will be applied to the entries below `cn=tim, o=Your Company`. If no propagating entry owner is found in the entries above `cn=tim, o=Your Company`, then the default entry owner is used.

Figure 33 on page 174 shows how to remove an `entryOwner` value from existing entry owner information:

```
ldapmdfy -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 33. Example of removing a single entryOwner Attribute value

In Figure 33 on page 174, 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 needs to 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 LDAPMDFY command to delete the entire attribute (all values) from the entry.

Figure 34 on page 174 shows an example of deleting an entry owner set from an entry.

```
ldapmdfy -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 34. Example of deleting an entry owner set from an entry

In Figure 34 on page 174, the existing entry owner set against `cn=jeff, cn=tim, o=Your Company` is removed. This means that the entry owner information will no longer apply to the entry. Instead, the first propagating entry owner set that is found in an entry above `cn=jeff, cn=tim, o=Your Company` will be applied to `cn=jeff, cn=tim, o=Your Company`. If no propagating entry owner set is found in the entries above `cn=jeff, cn=tim, o=Your Company`, then the default entry owner is used.

Creating a group for use in ACLs and entry owner settings

Sets of users can be grouped together in the directory by defining them as members of a group in the directory. A directory group, used for access control checking, is just another entry in the directory. A static, dynamic, or nested group entry can be used as a group on the **aclEntry** or **entryOwner** attributes. See Chapter 9, "Static, dynamic, and nested groups," on page 133 for more information on 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 if a user can perform the action requested.

Groups are added to access control information in just the same way as user entries are added to access control information. Figure 35 on page 175 shows how a group can be added to the `aclEntry` information in an existing access control specification for an entry. Figure 36 on page 175 shows how a group can be added as an `entryOwner` to an existing entry owner specification for an entry.

```
ldapmdfy -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 35. Example of adding a group to access control information

```
ldapmdfy -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 36. Example of adding a group to entry owner information

Chapter 11. Basic replication

Once the z/VM 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 will only accept updates from their master server.

If you need more advanced replication choices, see Chapter 12, "Advanced replication," on page 193.

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 mode. For more information about server operating modes, see LDAP Server Operational Mode in *z/VM: TCP/IP Planning and Customization*.

In z/VM LDAP, basic replication is supported in an LDBM backend. Basic replication is not performed for the SDBM or GDBM backends or for the schema entry.

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/VM LDAP servers have this capability. If the root DSE of a replica server does not contain the required capability, then the **ibm-entryuuid** attribute will not be replicated to that server, however, the entry and other attributes will be replicated.

Complex modify DN replication

Basic replication of Modify DN new superior operations will be 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/VM LDAP servers have this capability. If a replica server is not at a supported level, Modify DN new superior operations will fail until the replica is removed from the replica collection.

Password encryption and basic replication

To ensure data integrity and the correct working of the LDAP servers in the replication environment, the pwEncryption option in the configuration files for the servers involved in replication must be the same. If one of the servers involved in replication is a non-z/OS or non-z/VM server, then the administrator must select a pwEncryption method that is supported by both servers for correct operation of replication. If no encryption methods are common between the servers, then password encryption should not be used.

When replicating between a z/VM LDAP server and a non-z/OS or non-z/VM LDAP server and using **crypt** for password encryption, specify **pwCryptCompat off** in the backend section of the z/VM LDAP server configuration file. This setting indicates that the LDAP server should use the UTF-8 version of the crypt algorithm to encrypt passwords. When **userPassword** attribute values in **crypt** are replicated between z/VM and non-z/OS or non-z/VM LDAP servers, the password will be the same on both platforms and therefore it will be usable.

If using AES or DES encryption and the key is stored in an LDAPKEYS file and both of the servers involved in replication are z/VM LDAP servers, the same key label and data key must be present in both server's copy of the LDAPKEYS file. The AES or DES key label is specified in the LDAP server configuration files of both of the LDAP servers involved in replication.

Data encryption or hashing and basic replication

When encryption or hashing is configured in an LDBM 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 179 and "Configuring the replica" on page 184 for more information.

2. If the **pwEncryption** configuration option is set to **crypt** and replication is configured between a z/VM LDAP server and a non-z/VM or 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/VM and non-z/VM LDAP servers, the password is the same on both platforms and, therefore, is usable.

   If basic replication is configured between two z/VM LDAP servers or a z/VM LDAP server and a z/OS LDAP server, 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 the values are usable on the other server.

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 185 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. In order to reset the count, either the server needs to be restarted or the replica entry needs to be removed and added. See "Basic replication error log" on page 188 for more information about the set aside count.

Table 30. Replica entry schema definition (mandatory attributes)

Attribute	Description and example
replicaHost	This can be an IPv4 address, IPv6 address, or a hostname 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.hal.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 will be 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 30. 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 30 on page 179, when the replicating server receives and successfully finishes an update request, the update is also sent to `myMachine.hal.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 The Administrator DN and the Replica Server DN and Passwords in *z/VM: TCP/IP Planning and Customization* 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 then the default port of 389. Examples of adding a description and changing the port to 400 are shown in Table 31 on page 180, which describes the optional attributes of **replicaObject**.

Table 31. 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=HAL, c=US`

Table 31. Replica entry schema definition (optional attributes) (continued)	
Attribute	**Description and example**
`replicaBindMethod`	Identifies the bind method to be used. If it is specified, it must be set to `simple`. Example: `replicaBindMethod: simple`

Basic replication supports only simple authentication. SASL EXTERNAL, 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 188 for more information about error handling. These attributes are not in any object class, therefore, the **extensibleObject** object class must included in the replica entry when adding these attributes to the entry. Table 32 on page 181 describes these attributes.

Table 32. Additional optional replication attributes	
Attribute	**Description and example**
`ibm-slapdLog`	Specifies the file name of the basic replication error log. This must be a BFS file. 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 recommended. 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 will not occur. Example: `ibm-slapdLog: /home/replog/replica1.errlog`
`ibm-slapdReplMaxErrors`	Specifies the maximum number of basic replication errors that will be 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 LDBM

In LDBM, replica entries can be placed anywhere within the directory tree, although it is recommended 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 will not be 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 will be 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 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.hal.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 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:

```
ldapsrch -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 command can be used to display information about the status of replication to each replica server. See SMSG Interface to the LDAP Server in *z/VM: TCP/IP Planning and Customization* 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 which 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 6, "Administrative group and roles," on page 91 for more information.

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 (LDAPSRV command) to start the LDAP server in maintenance mode.

You can use the SMSG command to change from maintenance mode to normal mode while the LDAP server is running. The command can also be used to put a running server into maintenance mode. For example:

```
smsg ldapsrv maintmode on
```

turns maintenance mode on for the server whose user ID is LDAPSRV, and

```
smsg ldapsrv maintmode off
```

turns maintenance mode off (and normal mode on) for the same server.

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 will be required each time the schema is updated on the replicating server.

Replica servers must support LDAP 3.

Populating a replica

1. Either start the replica and replicating servers in maintenance mode or on each of these LDAP servers use the SMSG LDAPSRV MAINTMODE ON command to put these servers into maintenance mode.
2. Unload the replicating server's directory contents if there are any entries. For LDBM, use DS2LDIF (the ds2ldif utility); see DS2LDIF (ds2ldif utility) in *z/VM: TCP/IP Planning and Customization*).
3. You should make sure the schema for the replica server is the same as the schema for the replicating server.

   If the replica and replicating server are both z/VM 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 183.
6. Transport the LDIF file created in step "2" on page 183 to the replica server's location.
7. Load the LDIF file from step "6" on page 183 into the replica server. This can be done using an LDAP root administrator to run the **ldapadd** utility 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 SMSG *ldapsrv* 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/VM 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 hostname 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 hostname or IP address specified on the **listen** configuration option or the **-l** LDAP server command line parameter.
replicaPort	The port number that is specified on the **listen** configuration option or the **-l** LDAP server command line parameter.
replicaUseSSL	Use of **ldaps://** in the prefix of the `listen` configuration option or the **-l** LDAP server command line parameter corresponds to **TRUE** for **replicaUseSSL**; use of **ldap://** corresponds to **FALSE**.
replicaBindDn	masterServerDN or peerServerDN configuration option
replicaCredentials	masterServerPW or peerServerPW configuration option

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 Step 6. Create and Customize the LDAP Configuration File (DS CONF) in *z/VM: TCP/IP Planning and Customization*.

3. It is recommended that the `masterServerDN` or `peerServerDN` be a DN that is dedicated specifically to replication. It should not be used for any other operations.

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 The Administrator DN and the Replica Server DN and Passwords in *z/VM: TCP/IP Planning and Customization* for alternatives.

6. The **replicaCredentials** attribute will be 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 187 for information about securing a directory.

Changing a read-only replica to a master

When using read-only basic replication, it might become desirable to change one of the read-only replicas to be the master. Perhaps the machine where the replica server is installed is being upgraded, and 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 described in "Recovering from basic replication out-of-sync conditions " on page 190.
2. Use the SMSG LDAPSRV MAINTMODE ON 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 182 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. Remove all the files in the LDBM database directory. See the description of the **databaseDirectory** option in Step 6. Create and Customize the LDAP Configuration File (DS CONF) in *z/VM: TCP/IP Planning and Customization* 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 The Administrator DN and the Replica Server DN and Passwords in *z/VM: TCP/IP Planning and Customization* 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/VM LDAP peer replication server provides failover support. With this support, if a LDAP server fails, the peer replication server can take over the role of the failing LDAP server and it is then available to process LDAP operations.

A z/VM 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 183. The instructions in "Adding replica entries in LDBM" on page 181 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. For more information, see Step 6. Create and Customize the LDAP Configuration File (DS CONF) in *z/VM: TCP/IP Planning and Customization*.

Note: Usage of the **peerServerPW** configuration option is strongly discouraged in production environments. See The Administrator DN and the Replica Server DN and Passwords in *z/VM: TCP/IP Planning and Customization* 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 188 for more information.

When a conflict occurs, a notification will be sent to the console and server log.

Adding a peer replica to an existing server

For failover support, it 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/VM 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 The Administrator DN and the Replica Server DN and Passwords in *z/VM: TCP/IP Planning and Customization* 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 (use DS2LDIF) and adding the data to the new peer replica (use LDAPADD). See "Populating a replica" on page 183 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 will now be peer servers. See The Administrator DN and the Replica Server DN and Passwords in *z/VM: TCP/IP Planning and Customization* 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 The Administrator DN and the Replica Server DN and Passwords in *z/VM: TCP/IP Planning and Customization* 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 in *z/VM: TCP/IP Planning and Customization* 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/VM: TCP/IP User's Guide*), this time as if you were the client. The key database file 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 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 in *z/VM: TCP/IP Planning and Customization* 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 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.

The replication error log is used to correct basic replication in two ways:

- Use the error information to determine why replication failed.
- Use LDAPMDFY 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=USER01, o=hal,c=us to
  dceimgtd.hdl.hal.com:390, rc=32
# R004071 Entry cn=USER01, o=hal,c=us does not exist
  (ldbm_process_request:933)
# setting change aside.

dn: cn=USER01, o=hal,c=us
changetype: modify
modify: uid
uid: USER1
-
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 LDAPMDFY 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=USER01,
o=hal,c=us to dceimgtd.hdl.hal.com:390, rc=32
# R004071 Entry cn=USER01, o=hal,c=us does not exist
  (ldbm_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, use LDAPSRCH with a base of cn=localhost and a filter of objectClass=*. Otherwise, use LDAPSRCH 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 LDBM, the search must be repeated using each suffix in the search base.

  See *z/VM: TCP/IP User's Guide* for more information about LDAPSRCH.

- 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 using LDAPSRCH 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 a large number of entries, ensure that the region size for the replicating server is sufficient for replicating the entries to the replica. Entries on the replicating server are kept in memory during replication. If the region size is not sufficient, an out of memory condition can occur in the LDAP server. If possible, set the region size on the replicating server to 0M (or unlimited). If that cannot be done, set the region size to 14M (needed to run the LDAP server itself) plus 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. Issue SMSG LDAPSRV MAINTMODE ON on the replicating sever and on each of the replica servers to put them into maintenance mode.
2. 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 182 to create LDIF output containing the replica entries for each suffix in the backend.
3. Using a root administrator DN, run **ldapdelete** to remove the replica entries from the master. This resets the replication information in the replicating server.
4. Stop all the replica servers.
5. Clear out the directory on each replica server. Remove all the files in the LDBM database directory. See the description of the **databaseDirectory** option in Step 6. Create and Customize the LDAP Configuration File (DS CONF) in *z/VM: TCP/IP Planning and Customization* for more information about the location of these files.
6. Run an unload utility on the replicating server. Use DS2LDIF twice, once to unload the schema entry and a second time to unload the LDBM directory entries.
7. Start the replica servers in maintenance mode.
8. Using an administrator DN, run LDAPMDFY to load the schema unloaded from the replicating server onto each replica.
9. On each replica, use LDAPADD to load the directory data retrieved above from the replicating server. LDAPADD must be run using the administrator DN.
10. On each replica, load the directory data retrieved above from the replicating server. For LDBM, you must use **ldapadd**. The **ldapadd** utility must be run using a root administrator DN.
11. Using an administrator DN, run LDAPADD to add the replica entries unloaded in step "2" on page 190 back into the replicating server.

12. Issue SMSG LDAPSRV MAINTMODE OFF to take the replicating server and each replica out of maintenance mode.

Chapter 12. 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 off-loads 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 resides 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 need to 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/VM LDAP 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), and gateway.

Table 33. 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. **Notes:** 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.

Table 33. Server roles (continued)

Option	Description
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. For more information, see "Recovering from advanced replication errors" on page 249.

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, which 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 master's entire directory, or a subtree of the directory. In the following example, `Replica 2` contains a copy of the complete directory of `Master Server`, `Replica 1`, and `Replica 3` each contain a copy of a subtree of the `Master Server` directory.
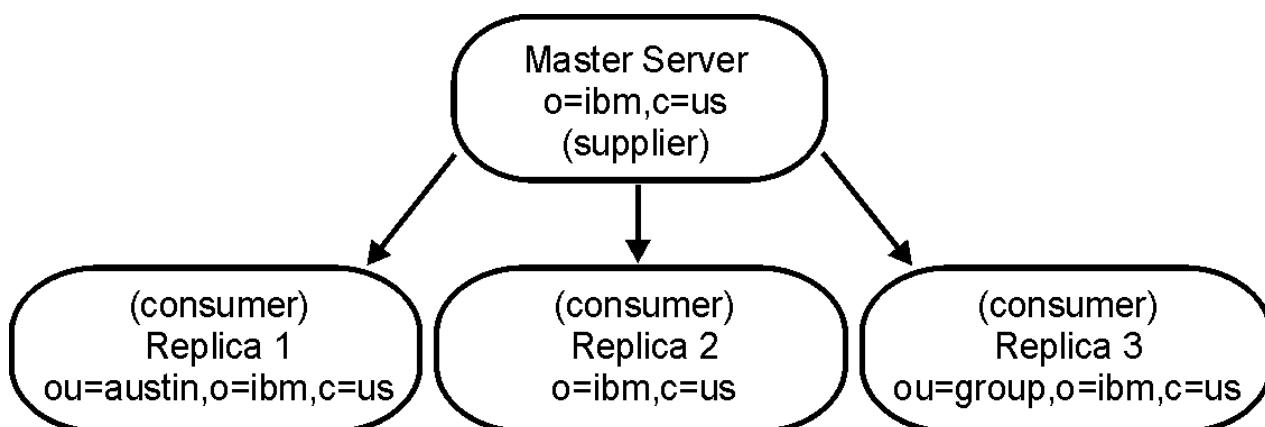
Figure 37. Master-replica replication

The relationship between two servers can also be described in terms of roles, either supplier or consumer. In the previous example, `Master Server` is a supplier to each of the replicas. Each replica in turn is a consumer of `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 off-loads 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. These replica servers are consumers of their respective forwarding servers. For example:
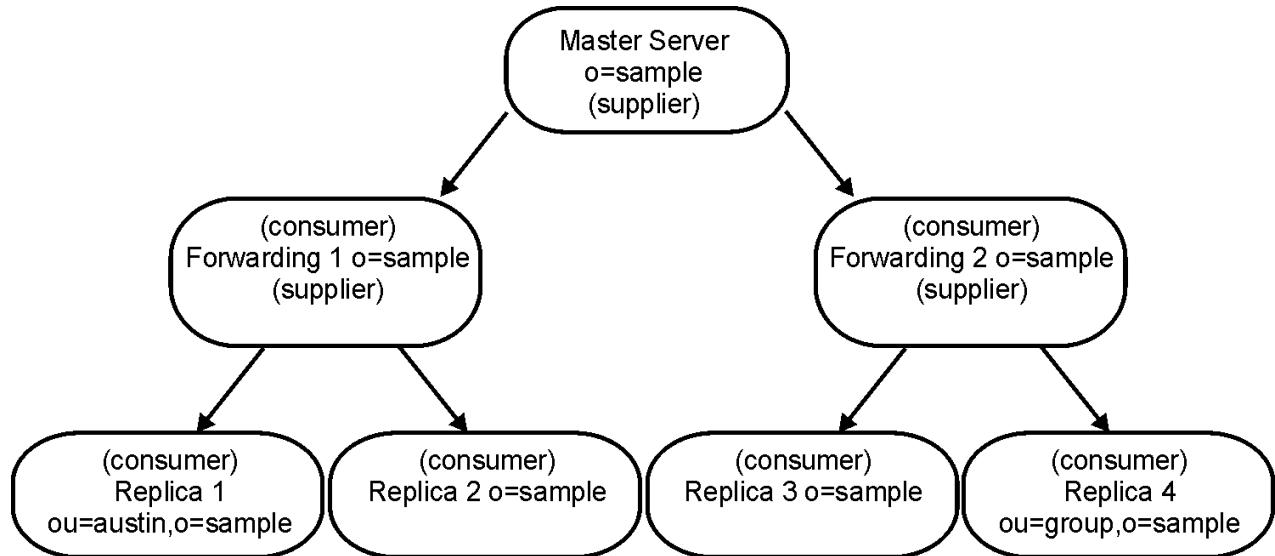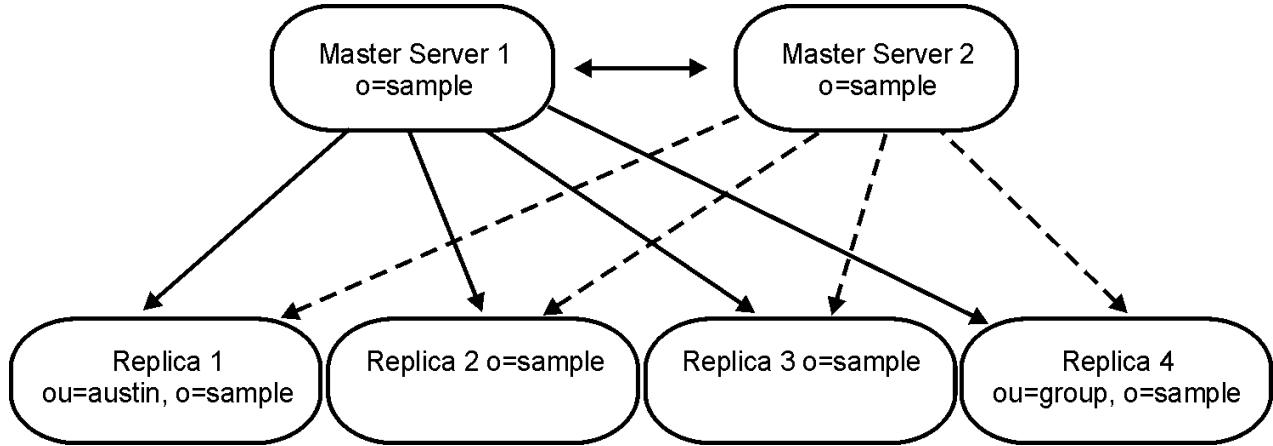
Figure 38. Cascading replication

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. for more information, see "Replication conflict resolution" on page 199.

Figure 39 on page 198 is an example of peer-to-peer replication:

Figure 39. Peer-to-peer replication
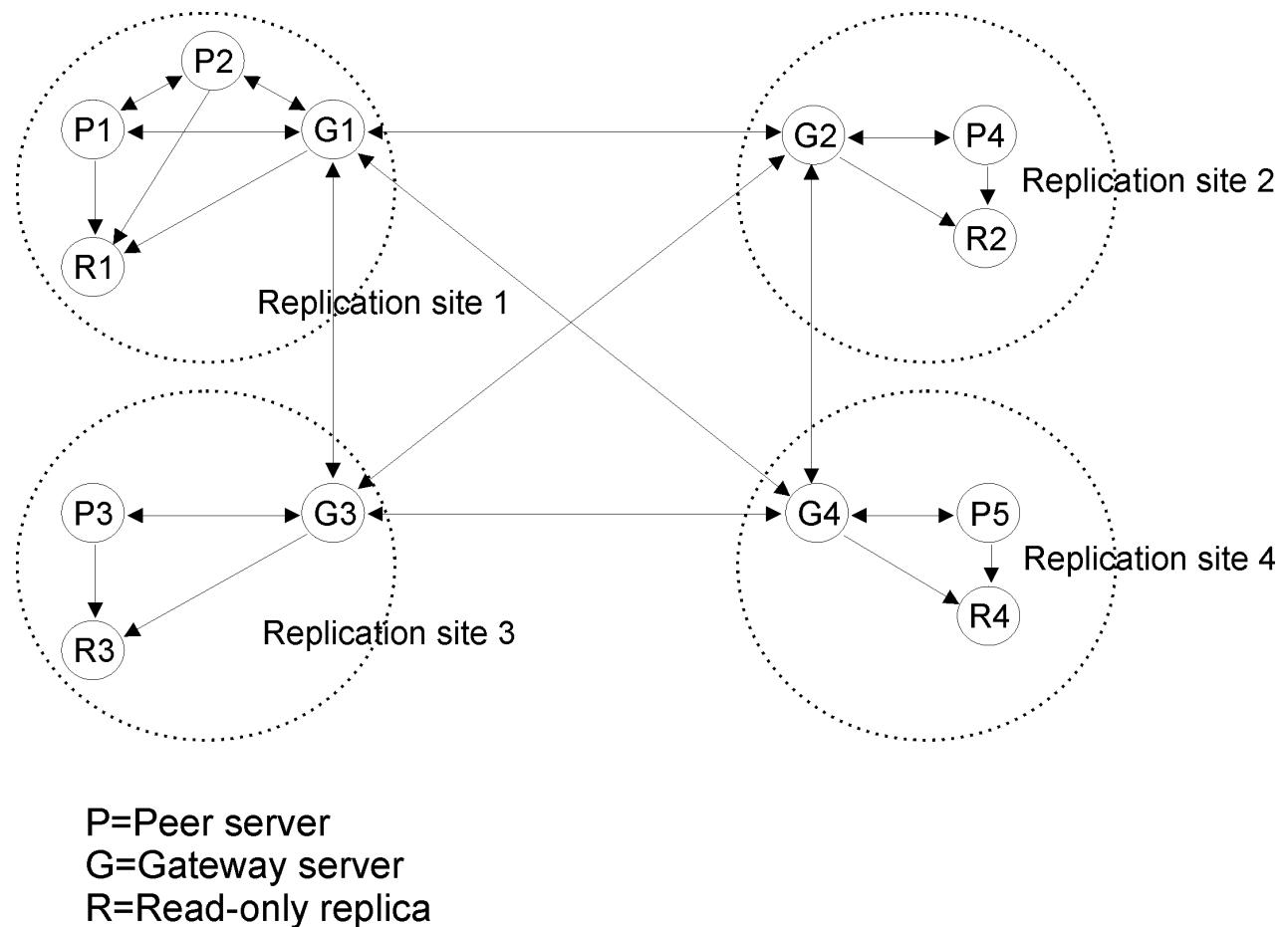
Gateway replication

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:

P=Peer server
G=Gateway server
R=Read-only replica

Figure 40. 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 240.

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 208.

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, the 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, to resynchronize the servers use the synchronization procedure in "Recovering from advanced replication errors" on page 249 .

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 LDBM backend, 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
databaseDirectory /var/ldap/cdbm
useAdvancedReplication on
```

Notes:

1. If **useAdvancedReplication on** is specified in the CDBM backend and basic replication entries with an objectclass of **replicaObject** exist in any configured LDBM backend, 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/VM LDAP server. If planning to use an advanced replication environment, all basic replication **replicaObject** entries must be removed from the LDBM backend.

2. If there are advanced replication entries in the LDBM backend 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.

3. The **masterServer, masterServerDN, masterServerPW, peerServerDN**, and **peerServerPW** configuration options are not allowed to be specified in any LDBM backend when the CDBM backend is configured and the **useAdvancedReplication** option is set to **on**. The **masterServer, masterServerDN, masterServerPW, peerServerDN**, and **peerServerPW** options are valid only 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 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
```

For more information about these entries and attribute values that affect advanced replication configuration, see CDBM backend configuration and policy entries in *z/VM: TCP/IP Planning and Customization*.

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. For the optional attribute value for the **ibm-replicationContext** objectclass, see Table 34 on page 201. 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:*. For more information about protecting the replication topology, see "Protecting replication topology entries" on page 238.

*Table 34. **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 separated by spaces. This list contains a list of servers that have update access to this replication context.
Example:
`ibm-replicaReferralURL: ldap://master1.hal.com:500 ldaps://master2.hal.com:636`

For the example in Table 34 on page 201, assume that a replication context of o=hal is used. When a client attempts an update operation on the consumer server under the o=hal 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**. For the optional and required attributes for the **ibm-replicaGroup** objectclass, see

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 affect how the replication topology is configured. For more information about protecting the replication topology, see .

Table 35. **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**. For the optional and required attributes for the **ibm-replicaSubentry** objectclass, see . A replica subentry identifies the role 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. For more information, see . There should be only one replica subentry created for a single server under a given replication context.

Table 36. **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 affected advanced replication configuration.   Example: `description: Represents the LDAP server (master1) under this replication context`

Table 36. **ibm-replicaSubentry** objectclass schema definition (optional and required attributes) (continued)
Attribute description and example
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 readded.   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. For more information about replication agreement entries, see "Replication agreements" on page 203.   Example: ``ibm-replicaServerID: supplier1``
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 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 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 36 on page 202, 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 resides.

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. For the required attributes for the **ibm-replicationAgreement** objectclass, see Table 37 on page 204. For the optional attributes for the **ibm-replicationAgreement** objectclass, see Table 38 on page 205. 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 37. **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. For more information about credential entries, see "Credentials entries" on page 206.

Example:

```
ibm-replicaCredentialsDN: cn=consumer1,cn=localhost
```

ibm-replicaURL

Specifies the LDAP URL of the consumer server. The LDAP URL syntax is fully documented in RFC 2255: The LDAP URL Format (www.ietf.org/rfc/rfc2255.txt?number=2255).

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. For more information about using SSL/TLS in an advanced replication environment, see "SSL/TLS and advanced replication" on page 242.

Example:

```
ibm-replicaURL: ldaps://consumer1.hal.com:500
```

For the examples in Table 37 on page 204, 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 located on host name consumer1.hal.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 38. **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. For more information about advanced replication scheduling, see "Schedule entries" on page 208.

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
```

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/VM LDAP server and 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. For additional information about using partial replication, see "Partial replication" on page 240.

Example:

```
ibm-replicationFilterDN: cn=filter,o=ibm
```

Table 38. *ibm-replicationAgreement* objectclass schema definition (optional attributes) (continued)
Attribute description and example
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. For more information about the **Cascading control replication** extended operation, see "Cascading control replication" on page 358. For more information about the **Control replication** extended operation, see "Control replication" on page 362.
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 server's 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 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 (ds2ldif utility) in *z/VM: TCP/IP Planning and Customization* for more information.

The objectclass of the entry specified in the **ibm-replicaCredentialsDN** attribute value in the replication agreement indicates the authentication method used by the supplier to authenticate with the consumer server. If the entry's objectclass is **ibm-replicationCredentialsSimple**, the supplier server uses a simple bind to authenticate to the consumer. For the required attributes of the **ibm-replicationCredentialsSimple** objectclass, see Table 39 on page 207. If the entry's objectclass is **ibm-replicationCredentialsExternal**, the supplier server performs a SASL EXTERNAL bind to the consumer server. For the optional attributes of the **ibm-replicationCredentialsExternal** objectclass, see Table 40 on page 207.

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. For more information about the consumer server credential entries, see "Consumer server entries" on page 219.

Table 39. **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 39 on page 207, 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 40. **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 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. For the acceptable formats for this attribute value, see sslKeyRingFile in *z/VM: TCP/IP Planning and Customization*. **Note:** If a value is specified for this attribute, it must be the same for all replication agreements in the server. The LDAP server supports having only one opened SSL/TLS key database file at a time. It is suggested that all SSL certificates that need to be used by the server be placed in one SSL/TLS key database file. 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 reside in the SSL/TLS key database file 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. For more about the **sslCertificate** option, see sslcertificate in *z/VM: TCP/IP Planning and Customization*. Example: `    ibm-replicaKeyLabel: EXTERNAL1`

*Table 40. **ibm-replicationCredentialsExternal** objectclass schema definition (optional attributes) (continued)*
Attribute description and example

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. Specify this attribute only 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 Byte 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 , 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 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 are not specified in the SASL EXTERNAL supplier server credentials entry, the default SSL configuration in the LDAP server configuration file is used.

For more information about using SSL/TLS in an advanced replication environment, see .

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**. For the schema definition of the **ibm-replicationWeeklySchedule** objectclass, see . 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 41. *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=hal`
ibm-scheduleTuesday Specifies the distinguished name (DN) of a daily replication schedule entry for Tuesday. Example: `ibm-scheduleTuesday: cn=tuesday,o=hal`
ibm-scheduleWednesday Specifies the distinguished name (DN) of a daily replication schedule entry for Wednesday. Example: `ibm-scheduleWednesday: cn=wednesday,o=hal`
ibm-scheduleThursday Specifies the distinguished name (DN) of a daily replication schedule entry for Thursday. Example: `ibm-scheduleThursday: cn=thursday,o=hal`

Table 41. **ibm-replicationWeeklySchedule** objectclass schema definition (optional attributes) (continued)
Attribute description and example
ibm-scheduleFriday Specifies the distinguished name (DN) of a daily replication schedule entry for Friday. Example: ``` ibm-scheduleFriday: cn=friday,o=hal ```
ibm-scheduleSaturday Specifies the distinguished name (DN) of a daily replication schedule entry for Saturday. Example: ``` ibm-scheduleSaturday: cn=saturday,o=hal ```
ibm-scheduleSunday Specifies the distinguished name (DN) of a daily replication schedule entry for Sunday. Example: ``` ibm-scheduleSunday: cn=sunday,o=hal ```

A daily replication schedule entry has an object class of **ibm-replicationDailySchedule**. For the schema definition of the **ibm-replicationDailySchedule** objectclass, see Table 42 on page 210. 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 42. **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 AM and then restarts at 9:45 AM and continues for the   rest of the day until 6:30 AM the following day. ```

Table 42. **ibm-replicationDailySchedule** *objectclass schema definition (optional attributes) (continued)*
Attribute description and example
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. For more information about the **Cascading control replication** extended operation, see "Cascading control replication" on page 358. For more information about the **Control replication** extended operation, see "Control replication" on page 362.
 	
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.

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. For more information about the **Cascading control replication** extended operation, see "Cascading control replication" on page 358. For more information about the **Control replication** extended operation, see "Control replication" on page 362.
 	
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 AM.

Table 42. **ibm-replicationDailySchedule** *objectclass schema definition (optional attributes) (continued)*
Attribute description and example
ibm-replicationTimesUTC A boolean (true or false) indicating whether the time values specified in the **ibm-replicationBatchStart** and **ibm-replicationImmediateStart** attributes are in GMT or local time. If set to true, GMT 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 , 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 AM each day. All future replication updates are queued.

- At 9:30 AM each day, advanced replication restarts because an **ibm-replicationImmediateStart** attribute value has been specified. Replication immediately starts and continues until the next day at 6:30 AM.

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 and the LDAP server starts at 7:00 AM, replication is suspended until 9:30 AM 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 reside under the **cn=configuration** suffix in the CDBM backend.

Note: The consumer server credentials entry differs from the supplier server credentials entry. For more information about the supplier server credentials entry, see "Credentials entries" on page 206.

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.

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 reside 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 43. **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 ```
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.   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 reside 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 (www.ietf.org/rfc/rfc2255.txt?number=2255).   If an update operation is done by a user other than the supplier server or any LDAP administrator with **Server Administration** control, this value is returned as one of the referral values.   For more information about referrals with advanced replication, see "Replication topology hints and tips" on page 237.   Example: ``` ibm-slapdMasterReferral: ldap://master1.ibm.com:500 ```

*Table 43. **ibm-slapdReplication** objectclass schema definition (required and optional attributes) (continued)*
Attribute description and example
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 attempt automatically 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 43 on page 213, 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 44. **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. Example: `ibm-slapdMasterDN: cn=supplier,cn=localhost`

*Table 44. **ibm-slapdSupplier** objectclass schema definition (required and optional attributes) (continued)*
Attribute description and example
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 reside 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 44 on page 214, 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, the administrator needs to 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 the LDAP administrator by querying the operational attributes in the replication agreement entries. For a description of the information available, see "Monitoring and diagnosing advanced replication problems" on page 246. 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. For more information about replication conflict resolution, see "Replication conflict resolution" on page 199.

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. For more information, see "Advanced replication maintenance mode" on page 239.

a. If using an existing subtree for advanced replication:

   i) 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:*.

   ii) Unload the entire subtree to an LDIF file by using the DS2LDIF utility. For additional information about the DS2LDIF utility, see DS2LDIF (ds2ldif utility) in *z/VM: TCP/IP Planning and Customization*).

   iii) For each server participating in the replication topology, add the unloaded entries to the server by using the LDAPADD utility. For more information about the LDAPADD utility, see Using the LDAP Client Utilities in *z/VM: TCP/IP User's Guide*.

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 utility. For more information about the LDAPADD utility, see Using the LDAP Client Utilities in *z/VM: TCP/IP User's Guide*.

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. For more information, see "Consumer server entries" on page 219.

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. For more information about schema replication, see "Replication of schema and password policy updates" on page 238.

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**. For more information, see "Replication contexts" on page 201.

   b. Replica group with an objectclass of **ibm-replicaGroup**. For more information, see "Replica groups" on page 201.

   c. Replica subentry with an objectclass of **ibm-replicaSubentry**. For more information, see "Replica subentries" on page 202.

   d. Replication agreement with an objectclass of **ibm-replicationAgreement**. For more information, see "Replication agreements" on page 203.

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. For more information about the LDAPEXOP utility, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

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 45. Topology setup	
Server	**Host name**
Server 1	server1.hal.com
Server 2	server2.hal.com
Server 3	server3.hal.com

Each LDAP server in Table 45 on page 217 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 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 entry's attribute **ibm-serverID**. The server ID is only allowed to be modified when there are no replica subentries defined in the server. For more information about the **ibm-slapdServerID** attribute value, see CDBM backend configuration and policy entries in *z/VM: TCP/IP Planning and Customization*.

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=hal,c=us`, the replication context might be:

```
dn: o=hal,c=us
objectclass: top
objectclass: organization
objectclass: ibm-replicationContext
o: hal
```

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=hal,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=hal,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=hal,c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: Peer2
ibm-replicaConsumerId: Peer2
ibm-replicaUrl: ldap://server2.hal.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.hal.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, for information about the **ibm-replicationCredentialsExternal** objectclass, see "Credentials entries" on page 206. 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=hal,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=hal,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 reside 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.
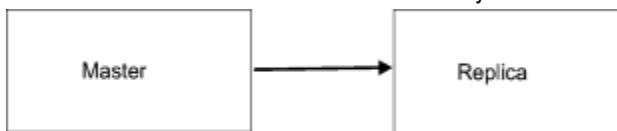
Figure 41. Master-replica topology

- The first step when building a topology is to define:

  1. **Replication context:** `o=hal,c=us`

  2. **Supplier(s):** LDAP server on `server1.hal.com:389` is the only supplier. The server ID is `Master`. It supplies updates to the LDAP server on `server2.hal.com:389`.

  3. **Consumer(s):** LDAP server on `server2.hal.com:389` is the only consumer. The server ID is `Replica`. It consumes updates from the LDAP server on `server1.hal.com:389`.

4. **Read-write server(s):** LDAP server on `server1.hal.com:389`, with ID `Master` is the only read-write server.

5. **Read-only server(s):** LDAP server on `server2.hal.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. For more information about the **ibm-slapdServerID** attribute value, see CDBM backend configuration and policy entries in *z/VM: TCP/IP Planning and Customization*.

1. **Server IDs:**

   – On the master server, apply the following modify using the LDAPMDFY utility. For additional information about the LDAPMDFY utility, see Using the LDAP Client Utilities in *z/VM: TCP/IP User's Guide*.

   ```
   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. For additional information about the LDAPADD utility, see Using the LDAP Client Utilities in *z/VM: TCP/IP User's Guide*. 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.hal.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 LDAPMDFY utility to modify the existing entry. For example:

   ```
   dn: o=hal,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=hal,c=us
   changetype: add
   objectclass: top
   objectclass: organization
   objectclass: ibm-replicationContext
   o: hal
   ```

2. Add the replica group entry using the LDAPADD utility. For example:

   ```
   dn: ibm-replicaGroup=default, o=hal,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=hal,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=hal,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=hal,c=us` subtree to the replica that consumes the changes. Therefore, there is only one agreement: From the master to the replica. Note 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=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Replica
ibm-replicaConsumerId: Replica
ibm-replicaUrl: ldap://server2.hal.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=hal,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.hal.com:389` meaning that the replica is listening on `server2.hal.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 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=hal,c=us
changetype: add
```

```
objectclass: top
objectclass: organization
objectclass: ibm-replicationContext
o: hal

dn: ibm-replicaGroup=default, o=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicaGroup
ibm-replicaGroup: default

dn: ibm-replicaServerId=Master,ibm-replicaGroup=default, o=hal,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=hal,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=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Replica
ibm-replicaConsumerId: Replica
ibm-replicaUrl: ldap://server2.hal.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=hal,c=us
description: Replication agreement from master to replica.
```

The LDIF is different if the replication context exists. Rather than:

```
dn: o=hal,c=us
changetype: add
objectclass: top
objectclass: organization
objectclass: ibm-replicationContext
o: hal
```

it is:

```
dn: o=hal,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.hal.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.hal.com -p 389 -D adminDN -w adminPW -op repltopology
  -rc o=hal,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=hal, c=us replication context. For

additional information about the LDAPEXOP utility, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

- After the LDAPADD and LDAPEXOP commands are performed successfully, the master-replica topology is ready. The master accepts updates on the o=hal,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.
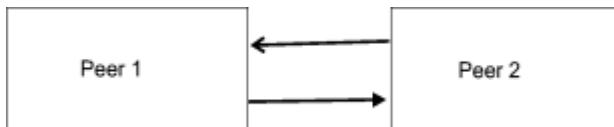
Figure 42. Peer-to-peer topology

- The first step when building a topology is to define:

    1. **Replication context:** o=hal,c=us

    2. **Supplier(s):** LDAP server on server1.hal.com:389 with server ID Peer1 supplies updates to the LDAP server with server ID Peer2 on server2.hal.com:389. LDAP server on server2.hal.com:389 with server ID Peer2 supplies updates to the LDAP server with server ID Peer1 on server1.hal.com:389.

    3. **Consumer(s):** LDAP server with server ID Peer2 on server2.hal.com:389 consumes updates from LDAP server with server ID Peer1 on server1.hal.com:389. LDAP server with Server ID Peer1 on server1.hal.com:389 consumes updates from LDAP server with server ID Peer2 on server2.hal.com:389.

    4. **Read-write server(s):** LDAP servers Peer1 and Peer2 on server1.hal.com and server2.hal.com are read-write servers.

    5. **Read-only server(s):** There are no read-only servers in this topology.

Configuration changes: Some CDBM changes need to 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 219.

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. For more information about the **ibm-slapdServerID** attribute value, see CDBM backend configuration and policy entries in *z/VM: TCP/IP Planning and Customization*.

    1. **Server IDs:**

        – On the Peer1 server (server1.hal.com), apply the following modify using the LDAPMDFY utility. For additional information about the LDAPMDFY utility, see Using the LDAP Client Utilities in *z/VM: TCP/IP User's Guide*.

        ```
        dn: cn=configuration
        changetype: modify
        replace: ibm-slapdserverid
        ibm-slapdserverid: Peer1
        ```

        – On the Peer2 server (server2.hal.com), apply the following modify using the LDAPMDFY 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. For additional information about the LDAPADD utility, see Using the LDAP Client Utilities in *z/VM: TCP/IP User's Guide*.

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.hal.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.hal.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 LDAPMDFY utility to modify the existing entry. For example:

     ```
     dn: o=hal,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=hal,c=us
     changetype: add
     objectclass: top
     objectclass: organization
     objectclass: ibm-replicationContext
     o: hal
     ```

  2. Add the replica group entry using the LDAPADD utility. For example:

     ```
     dn: ibm-replicaGroup=default, o=hal,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 need to 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=hal,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=hal,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=hal,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=hal,c=us` subtree to `Peer2`, that consumes the changes. `Peer2` also accepts updates from clients on the `o=hal,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=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Peer2
ibm-replicaConsumerId: Peer2
ibm-replicaUrl: ldap://server2.hal.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=hal,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=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Peer1
ibm-replicaConsumerId: Peer1
ibm-replicaUrl: ldap://server1.hal.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=hal,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=hal,c=us
changetype: add
```

```
objectclass: top
objectclass: organization
objectclass: ibm-replicationContext
o: hal

dn: ibm-replicaGroup=default, o=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicaGroup
ibm-replicaGroup: default

dn: ibm-replicaServerId=Peer1,ibm-replicaGroup=default, o=hal,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=hal,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=hal,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=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Peer2
ibm-replicaConsumerId: Peer2
ibm-replicaUrl: ldap://server2.hal.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=hal,c=us
description: Replication agreement from Peer1 to Peer2.

dn: cn=Peer1, ibm-replicaServerId=Peer2,ibm-replicaGroup=default,o=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Peer1
ibm-replicaConsumerId: Peer1
ibm-replicaUrl: ldap://server1.hal.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=hal,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.hal.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.hal.com -p 389 -D  adminDN -w adminPW -op repltopology
 -rc o=hal,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=hal,c=us replication context. For additional information about the LDAPEXOP utility, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

- 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.
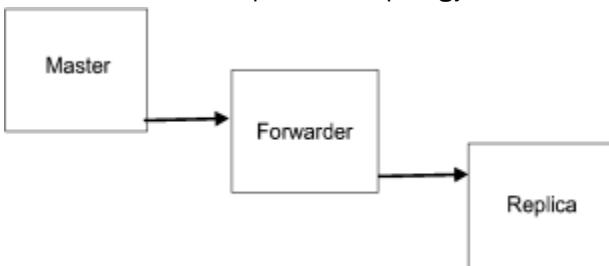
Figure 43. Master-forwarder-replica topology

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 needs to have one more server included: `server3.hal.com`

- The first step when building a topology is to define:

  1. **Replication context:** `o=hal,c=us`

  2. **Supplier(s):** LDAP server on `server1.hal.com:389` with server ID `Master` supplies updates to the LDAP server with server ID `Forwarder` on `server2.hal.com:389`. LDAP server on `server2.hal.com:389` with server ID `Forwarder` supplies updates to the LDAP server with server ID `Replica` on `server3.hal.com:389`.

  3. **Consumer(s):** LDAP server with server ID `Forwarder` on `server2.hal.com:389` consumes updates from LDAP server with server ID `Master` working on `server1.hal.com:389`. LDAP server with server ID `Replica` on `server3.hal.com:389` consumes updates from the LDAP server with server ID `Forwarder` working on `server2.hal.com:389`.

  4. **Read-write server(s):** LDAP server on `server1.hal.com:389`, with ID `Master` is the only read-write server.

  5. **Read-only server(s):** LDAP server on `server1.hal.com:389`, with IDs, `Forwarder` and `Replica` are read-only.

  Configuration changes: Some configuration changes need to 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 219. 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 223.

  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. For more information about the **ibm-slapdServerID** attribute value, see CDBM backend configuration and policy entries in *z/VM: TCP/IP Planning and Customization*.

  1. **Server IDs:**

- On the `Master` server (`server1.hal.com`), apply the following modify using the LDAPMDFY utility. For additional information about the LDAPMDFY utility, see Using the LDAP Client Utilities in *z/VM: TCP/IP User's Guide*.

```
dn: cn=configuration
changetype: modify
replace: ibm-slapdserverid
ibm-slapdserverid: Master
```

- On the `Forwarder` server (`server2.hal.com`), apply the following modify using the LDAPMDFY utility.

```
dn: cn=configuration
changetype: modify
replace: ibm-slapdserverid
ibm-slapdserverid: Forwarder
```

- On the `Replica` server (`server3.hal.com`), apply the following modify using the LDAPMDFY 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. For additional information about the LDAPADD utility, see Using the LDAP Client Utilities in *z/VM: TCP/IP User's Guide*.

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.hal.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.hal.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 LDAPMDFY utility to modify the existing entry. For example:

```
dn: o=hal,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=hal,c=us
changetype: add
objectclass: top
objectclass: organization
```

```
objectclass: ibm-replicationContext
o: hal
```

2. Add the replica group entry using the LDAPADD utility. For example:

```
dn: ibm-replicaGroup=default, o=hal,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=hal,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=hal,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=hal,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=hal,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=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Forwarder
ibm-replicaConsumerId: Forwarder
```

```
ibm-replicaUrl: ldap://server2.hal.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=hal,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=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Replica
ibm-replicaConsumerId: Replica
ibm-replicaUrl: ldap://server3.hal.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=hal,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=hal,c=us
changetype: add
objectclass: top
objectclass: organization
objectclass: ibm-replicationContext
o: hal

dn: ibm-replicaGroup=default, o=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicaGroup
ibm-replicaGroup: default

dn: ibm-replicaServerId=Master,ibm-replicaGroup=default, o=hal,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=hal,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=hal,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=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Forwarder
ibm-replicaConsumerId: Forwarder
ibm-replicaUrl: ldap://server2.hal.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=hal,c=us
description: Replication agreement from Master to Forwarder

dn: cn=Replica, ibm-replicaServerId=Forwarder,ibm-replicaGroup=default,o=hal,c=us
changetype: add
objectclass: top
```

```
objectclass: ibm-replicationAgreement
cn: Replica
ibm-replicaConsumerId: Replica
ibm-replicaUrl: ldap://server3.hal.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=hal,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.hal.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.hal.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.hal.com -p 389 -D adminDN -w adminPW -op repltopology -rc
  o=hal,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=hal,c=us replication context. For additional information about the LDAPEXOP utility, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

- 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 need to 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 resides 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 44. 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 server's server ID and the subentry's **ibm-replicationServerIsMaster** attribute is set to `true`. For more information about replica subentries, see "Replica subentries" on page 202.

- The first step when building a topology is to define:

  1. **Replication context:** `o=hal,c=us`

  2. **Supplier(s):** LDAP server on `server1.hal.com:389` with server ID `Gateway1` supplies updates to the LDAP server with server ID `Gateway2` on `server2.hal.com:389`. LDAP server on `server2.hal.com:389` with server ID `Gateway2` supplies updates to the LDAP server with server ID `Gateway1` on `server1.hal.com:389` and to the LDAP server with server ID `Replica` on `server3.hal.com:389`.

  3. **Consumer(s):** LDAP server with server ID `Gateway2` on `server2.hal.com:389` consumes updates from LDAP server with server ID `Gateway1` on `server1.hal.com:389`. LDAP server with Server ID `Gateway1` on `server1.hal.com:389` consumes updates from LDAP server with server ID `Gateway2` on `server2.hal.com:389`. LDAP server with server ID `Replica` on `server3.hal.com:389` consumes updates from LDAP server with server ID `Gateway2` on `server2.hal.com:389`.

  4. **Read-write server(s):** LDAP servers `Gateway1` and `Gateway2` on `server1.hal.com` and `server2.hal.com` are read-write servers.

  5. **Read-only server(s):** LDAP server `Replica` on `server3.hal.com` is a read-only server.

  Configuration changes: Some configuration changes need to 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 219. 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 223. 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 227.

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. For more information about the **ibm-slapdServerID** attribute value, see CDBM backend configuration and policy entries in *z/VM: TCP/IP Planning and Customization*.

1. **Server IDs:**

   – On the `Gateway1` server (`server1.hal.com`), apply the following modify using the LDAPMDFY utility. For additional information about the LDAPMDFY utility, see Using the LDAP Client Utilities in *z/VM: TCP/IP User's Guide*.

   ```
   dn: cn=configuration
   changetype: modify
   replace: ibm-slapdserverid
   ibm-slapdserverid: Gateway1
   ```

   – On the `Gateway2` server (`server2.hal.com`), apply the following modify using the LDAPMDFY utility.

   ```
   dn: cn=configuration
   changetype: modify
   replace: ibm-slapdserverid
   bm-slapdserverid: Gateway2
   ```

   – On the `Replica` server (`server3.hal.com`), apply the following modify using the LDAPMDFY 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. For additional information about the LDAPADD utility, see Using the LDAP Client Utilities in *z/VM: TCP/IP User's Guide*.

   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.hal.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.hal.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.hal.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 LDAPMDFY utility to modify the existing entry. For example:

     ```
     dn: o=hal,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=hal,c=us
     changetype: add
     objectclass: top
     objectclass: organization
     objectclass: ibm-replicationContext
     o: hal
     ```

  2. Add the replica group entry using the LDAPADD utility. For example:

     ```
     dn: ibm-replicaGroup=default, o=hal,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=hal,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=hal,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=hal,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=hal,c=us` subtree to `Gateway2`. `Gateway2` supplies updates made to the `o=hal,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=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Gateway2
ibm-replicaConsumerId: Gateway2
ibm-replicaUrl: ldap://server2.hal.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=hal,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=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Gateway1
ibm-replicaConsumerId: Gateway1
ibm-replicaUrl: ldap://server1.hal.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=hal,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=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Replica
ibm-replicaConsumerId: Replica
ibm-replicaUrl: ldap://server3.hal.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=hal,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=hal,c=us
changetype: add
objectclass: top
objectclass: organization
objectclass: ibm-replicationContext
o: hal

dn: ibm-replicaGroup=default, o=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicaGroup
ibm-replicaGroup: default

dn: ibm-replicaServerId=Gateway1,ibm-replicaGroup=default, o=hal,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=hal,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=hal,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=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Gateway2
ibm-replicaConsumerId: Gateway2
ibm-replicaUrl: ldap://server2.hal.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=hal,c=us
description: Replication agreement from Gateway1 to Gateway2.

dn: cn=Gateway1,ibm-replicaServerId=Gateway2,ibm-replicaGroup=default,o=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Gateway1
ibm-replicaConsumerId: Gateway1
ibm-replicaUrl: ldap://server1.hal.com:389
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=hal,c=us
description: Replication agreement from Gateway2 to Gateway1.

dn: cn=Replica, ibm-replicaServerId=Gateway2,ibm-replicaGroup=default,o=hal,c=us
changetype: add
objectclass: top
objectclass: ibm-replicationAgreement
cn: Replica
ibm-replicaConsumerId: Replica
ibm-replicaUrl: ldap://server3.hal.com:389
```

```
ibm-replicaCredentialsDN: cn=ReplicaBindCredentials, o=hal,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.hal.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.hal.com -p 389 -D adminDN -w adminPW -op repltopology
 -rc o=hal,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=hal,c=us` replication context. For more information about the LDAPEXOP utility, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

- 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 which then forwards them 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 need to 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 need to 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 need to supply to each other. There has to 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. for more information about replication contexts, see "Replication contexts" on page 201.

   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. For more information about consumer server entries, see "Consumer server entries" on page 212.

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. For more information about the **referral** configuration option, see Step 6. Create and Customize the LDAP Configuration File (DS CONF) in *z/VM: TCP/IP Planning and Customization*.

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 are already synchronized.

Note: The **ldapdiff** utility available on z/OS and other platforms facilitates the synchronization of schema and directory trees between servers. You can use the utility to synchronize the schema between z/VM LDAP servers. For more information about this utility, see *z/OS IBM Tivoli Directory Server Administration and Use*.

Before configuring replication of password policy and other entries in the **cn=ibmpolicies** suffix, verify that the entries are already synchronized. You can use the **ldapdiff** utility to synchronize the entries. 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 backend. For information about configuring advanced replication, see "Advanced replication configuration examples" on page 217; 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 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. For more information about the **ibm-slapdReplRestrictedAccess** attribute, CDBM backend configuration and policy entries in *z/VM: TCP/IP Planning and Customization*.

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. For more information about advanced replication maintenance mode, see "Advanced replication maintenance mode" on page 239.
2. Bind to the supplier server as an LDAP administrator. See Chapter 6, "Administrative group and roles," on page 91 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. For more information about advanced replication maintenance mode, see "Advanced replication maintenance mode" on page 239.

On the consumer server, perform the following steps to remove the replication topology entries:

1. Ensure that the consumer server is in maintenance mode. For more information about advanced replication maintenance mode, see "Advanced replication maintenance mode" on page 239.

2. Bind to the consumer server as an LDAP administrator. See Chapter 6, "Administrative group and roles," on page 91 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. For more information about advanced replication maintenance mode, see "Advanced replication maintenance mode" on page 239.

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. For more information about the **replicateOperationalAttributes** and **Server Administration** controls, see Appendix B, "Supported server controls," on page 345.

Consumer servers under maintenance mode continues 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 (LDAPSRV command) to start the LDAP server in maintenance mode. The LDAP server MAINTMODE operator 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:

```
smsg ldapsrv maintmode state
```

where `state` can be **on** to turn maintenance mode on 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 6, "Administrative group and roles," on page 91 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. For more information, see "Replication agreements" on page 203. A replication filter entry has a structural objectclass of **ibm-replicationFilter**. For the required attribute values for the **ibm-replicationFilter** objectclass., see Table 46 on page 240. 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. For more information about the LDAPEXOP utility, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

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. For the required attribute values for the format of the **ibm-replicationFilterAttr** attribute values, see Table 46 on page 240.

Table 46. **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 46. **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 server's 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 server's 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)
 	
Notes:
 	
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 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 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.

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 server's certificate in its key database file.

For more information, see Setting up for SSL/TLS in *z/VM: TCP/IP Planning and Customization*.

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 as if you were the client. For more information on the **gskkyman** utility, see SSL Certificate Management in *z/VM: TCP/IP User's Guide*. Receive the replica's 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.
   - 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. For information about the **ibm-replicaURL** attribute value, see Table 37 on page 204.

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. For more information about SASL EXTERNAL credentials entries, see Table 40 on page 207.

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 command is used to allow the LDAP administrator to easily query the current state of all configured replication contexts and replication agreements in the LDAP server. This command allows the LDAP administrator to verify that the advanced replication topologies are configured properly and are working as expected. For more information about the LDAP server **DISPLAY REPLICAS** operator command, see SMSG Interface to the LDAP Server in *z/VM: TCP/IP Planning and Customization*.

If the LDAP server **DISPLAY REPLICAS** operator 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. For more information, see "Monitoring and diagnosing advanced replication problems" on page 246.

Command line tasks for managing replication

This topic discusses 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 99 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. For more information about the LDAPEXOP utility, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

Table 47 on page 244 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 47. 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, the 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. For more information about the **Cascading control replication** extended operation, see "Cascading control replication" on page 358.
controlqueue	**Control replication queue** extended operation	This extended operation can skip one or all pending changes in the advanced replication queue. For more information about the **Control replication queue** extended operation, see "Control replication queue" on page 365.

LDAPEXOP operation	LDAPEXOP description	Overview
Table 47. Description of advanced replication extended operations on the LDAPEXOP utility (continued)		
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. For more information about the **Control replication** extended operation, see "Control replication" on page 362.
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. For more information about the **Control replication error log** extended operation, see "Control replication error log" on page 363.
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 6, "Administrative group and roles," on page 91 for more information about administrative role authority. See "Quiesce or unquiesce context" on page 370 for more information about the **Quiesce or unquiesce context** extended operation.
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. For more information about the **Replication topology** extended operation, see "Replication topology" on page 371.

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.

```
ldapsrch -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. For more information about replica subentries, see "Replica subentries" on page 202.
- 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. For more information about replication agreement entries, see "Replication agreements" on page 203.

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**. For the operational attributes for the **ibm-replicationContext** objectclass, see Table 48 on page 246. For the operational attributes for the **ibm-replicationAgreement** objectclass, see Table 49 on page 247.

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. For more information about the LDAPEXOP utility, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

Table 48. ***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.

For the optional non-operational attribute for the **ibm-replicationContext** objectclass, see Table 34 on page 201.

*Table 49. **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. For more information about the **ibm-slapdReplMaxErrors** attribute value, CDBM backend configuration and policy entries in *z/VM: TCP/IP Planning and Customization*.
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**. For more information about the **ibm-slapdReplMaxErrors** attribute value, CDBM backend configuration and policy entries in *z/VM: TCP/IP Planning and Customization*.
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**. For more information about the **ibm-slapdReplMaxErrors** attribute value, CDBM backend configuration and policy entries in *z/VM: TCP/IP Planning and Customization*. 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. For more information about the LDAPEXOP utility, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*. 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 200504407202221Z 68 1 170814 add cn=entry-85,o=HAL,c=US failureId: 1 timestamp: 200504407202221Z returnCode: 68 numOfAttempts: 1 changeId: 170814 operation: add entryDn: cn=entry-85,o=HAL,c=US ```

*Table 49. **ibm-replicationAgreement** operational attributes (continued)*
Attribute and description
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. For more information about replication schedule entries, see "Schedule entries" on page 208.

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:
 	
<pre>ibm-replicationLastResult: 20050412140436Z 19 81 add cn=testpendingchange,o=hal,c=us
 	
timestamp: 20050412140436Z
changeId: 19
returnCode: 81
operation: add
entryDn: cn=testpendingchange,o=hal,c=us</pre>

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 once 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.

*Table 49. **ibm-replicationAgreement** operational attributes (continued)*
Attribute and description

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=hal,c=us```
 ```changeId: 19```
 ```operation: add```
 ```entryDn: cn=test1,o=hal,c=us```

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. For the steps on how to recover from out of sync conditions between supplier and consumer servers, see "Recovering from advanced replication errors" on page 249.   • **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. For more information about replication schedule entries, see "Schedule entries" on page 208.

For the required non-operational attributes for the **ibm-replicationAgreement** objectclass, see Table 37 on page 204. For the optional non-operational attributes for the **ibm-replicationAgreement** objectclass, see Table 38 on page 205.

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. For more information about the **cn=Replication,cn=configuration** entry, CDBM backend configuration and policy entries in *z/VM: TCP/IP Planning and Customization*.

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 6, "Administrative group and roles," on page 91 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 resides.

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 resides 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. For more information about the **ibm-slapdMaxPendingChangesDisplayed** attribute, CDBM backend configuration and policy entries in *z/VM: TCP/IP Planning and Customization*. For more information about the **ibm-replicationPendingChanges** operational attribute, see Table 49 on page 247.

3. The **ibm-replicationFailedChanges** operational attribute in the replication agreement shows each of the failed changes, including the failure ID. For more information about the **ibm-replicationFailedChanges** operational attribute, see Table 49 on page 247.

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. For more information about the LDAPEXOP utility, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

When the latest and all previous failures are understood, the 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 resides. For more information about the **ibm-slapdReplMaxErrors** attribute, CDBM backend configuration and policy entries in *z/VM: TCP/IP Planning and Customization*.

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. For more information about the LDAPEXOP utility, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

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. For more information about the LDAPEXOP utility, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

4. If there are multiple failed and pending replication changes, the entire replication context where the replication agreement entry resides 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. For more information about the **Cascading control replication** extended operation, see "Cascading control replication" on page 358. For more information about the LDAPEXOP utility, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

   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. For more information about the **Control replication** extended operation, see "Control replication" on page 362.

   c. Use the **ldapdiff** utility or manually compare the replication contexts on each of the servers within the replication context.

   d. Delete all failed replication changes for all replication agreements by using the **Control replication error log** extended operation on the LDAPEXOP utility. For more information about the **Control replication error log** extended operation, see "Control replication error log" on page 363.

   e. Skip all pending replication changes by using the **Control replication queue** extended operation on the LDAPEXOP utility. For more information about the **Control replication queue** extended operation, see "Control replication queue" on page 365.

   f. Resynchronize the replication context manually or by using 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 6, "Administrative group and roles," on page 91 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 219. This example assumes the **ibm-slapdReplMaxErrors** attribute value in the **cn=Replication,cn=configuration** entry is set to one.

The LDAP administrator periodically monitors the replication status of the replication agreement in the o=hal,c=us replication context by querying the replication agreement operational attribute values. For more information about the replication agreement operational attributes, see Table 49 on page 247.

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 LDAPSRCH command to retrieve the replication agreement entry. For more information about the LDAPSRCH utility, see Using the LDAP Client Utilities in *z/VM: TCP/IP User's Guide*.

```
ldapsrch -p 389 –h server1.hal.com -D adminDn -w adminPw -b o=hal,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 LDAPSRCH command returns the following entry:

```
cn=Replica, ibm-replicaServerId=Master, ibm-replicaGroup=default, o=hal, c=us
objectclass=top
objectclass=ibm-replicationAgreement
ibm-replicaconsumerid=Replica
ibm-replicaurl=ldap://server1.hal.com:389
ibm-replicacredentialsdn=cn=ReplicaBindCredentials,o=hal, c=us
description=Replication agreement from master to replica
cn=Replica
ibm-replicationonhold=FALSE
ibm-replicationstate=retrying
ibm-replicationpendingchanges=46 modify OU=SUB,O=HAL,C=US
ibm-replicationpendingchangecount=1
ibm-replicationnexttime=19000101000000
ibm-replicationlastresultadditional=R004071 DN 'OU=SUB,O=HAL,C=US' does not exist
 (ldbm_process_request:406)
ibm-replicationlastresult=20090206145054Z 46 32 modify OU=SUB,O=HAL,C=US
ibm-replicationlastfinishtime=20090206144954Z
ibm-replicationlastchangeid=45
ibm-replicationlastactivationtime=20090206144354Z
ibm-replicationfailedchanges=12 20090206144954Z 32 1 45 add cn=entry,ou=sub,o=hal,c=us
ibm-replicationfailedchangecount=1
ibm-replicationchangeldif=
dn: ou=sub,o=hal,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 the last failure is a modify of the ou=sub,o=hal,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=hal,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=hal,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=hal,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=hal,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. For more information about the LDAPEXOP utility, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

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.hal.com -D adminDn -w adminPw -op controlreplerr
 -ra "cn=Replica, ibm-replicaServerId=Master, ibm-replicaGroup=default,
 o=hal,c=us" -show 12
```

The LDAPEXOP command returns the following:

```
dn: cn=entry,ou=sub,o=hal,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
```

The 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, the LDAP administrator has chosen to resynchronize the replication context manually.

Before you 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. For more information about the LDAPEXOP utility, see LDAPEXOP (ldapexop utility) in *z/VM: TCP/IP Planning and Customization*.

The following LDAPEXOP command quiesces the o=hal,c=us replication context on the master and replica server in the replication topology:

```
ldapexop -p 389 -h server1.hal.com -D adminDn -w adminPw -op cascrepl -action
 quiesce -rc "o=hal,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=hal,c=us. The cn=Replica, ibm-replicaServerId=Master, ibm-replicaGroup=default, o=hal, c=us is the only replication agreement within the o=hal,c=us replication context so that it is the only agreement that is suspended.

```
ldapexop -p 389 -h server1.hal.com -D adminDn -w adminPw -op controlrepl
 -action suspend -rc "o=hal,c=us"
```

Use LDAPSRCH against both servers using a baseDN of o=hal,c=us. This action should verify that ou=sub,o=hal,c=us does not exist on the consumer server, explaining why the addition of the child entry cn=entry,ou=sub,o=hal,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=hal, c=us:

```
ldapexop -p 389 –h server1.hal.com -D adminDn -w adminPw -op controlreplerr
 -delete all -ra "cn=Replica, ibm-replicaServerId=Master, ibm-replicaGroup=default,
 o=hal, c=us"
```

The following LDAPEXOP command skips (deletes) all pending replication changes from the replication queue:

```
ldapexop -p 389 –h server1.hal.com -D adminDn -w adminPw -op controlqueue
 –skip all -ra "cn=Replica, ibm-replicaServerId=Master, ibm-replicaGroup=default,
 o=hal, c=us"
```

To synchronize the o=hal,c=us replication context on the master and replica servers, run DS2LDIF against the supplier server specifying a subtree (-s option) of ou=sub,o=hal,c=us, and then run LDAPMDFY against the consumer server using the output LDIF file from DS2LDIF. The LDAPMDFY -k option is required to allow updates because the server is quiesced.

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=hal, c=us:

```
ldapexop –p 389 –h server1.hal.com –D adminDn –w adminPw –op controlrepl
 –action resume –ra "cn=Replica, ibm-replicaServerId=Master, ibm-
replicaGroup=default,
 o=hal, c=us"
```

The following LDAPEXOP command unquiesces the replication context o=hal,c=us on all servers in the replication topology:

```
ldapexop –p 389 –h server1.hal.com –D adminDn –w adminPw –op cascrepl
 –action unquiesce –rc "o=hal,c=us"
```

The current replication status from the master to the replica can be determined by using the following LDAPSRCH command to retrieve the replication agreement entry:

```
ldapsrch -p 389 –h server1.hal.com -D adminDn -w adminPw -b "o=hal,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 LDAPSRCH command returns the following entry:

```
cn=Replica, ibm-replicaServerId=Master, ibm-replicaGroup=default, o=hal, c=us
objectclass=top
objectclass=ibm-replicationAgreement
ibm-replicaconsumerid=Replica
ibm-replicaurl=ldap://server2.hal.com:389
ibm-replicacredentialsdn=cn=ReplicaBindCredentials,o=hal, 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 13. Alias

Alias support provides a means for an 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 LDAP department:

```
"ou=LDAPZOS,o=HAL"
```

The alias entry points to the actual LDAP department:

```
"ou=DEPT9K,ou=Holbrook,o=HAL_US,o=HAL"
```

This provides easier access to the entries of the LDAP developers, using public names such as:

```
"cn=kmorg,ou=LDAPZOS,o=HAL"
```

This name is dereferenced during search to:

```
"cn=kmorg,ou=Dept9K,ou=Holbrook,o=HAL_US,o=HAL
```

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=HAL
objectclass: organizationalUnit
objectclass: aliasObject
ou: LDAPZOS
aliasedobjectname: ou=Dept9K,ou=Holbrook,o=HAL_US,o=HAL
```

or

```
dn: ou=LDAPZOS,o=HAL
objectclass: alias
objectclass: extensibleobject
ou: LDAPZOS
aliasedobjectname: ou=Dept9K,ou=Holbrook,o=HAL_US,o=HAL
```

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 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 dereferenced 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.
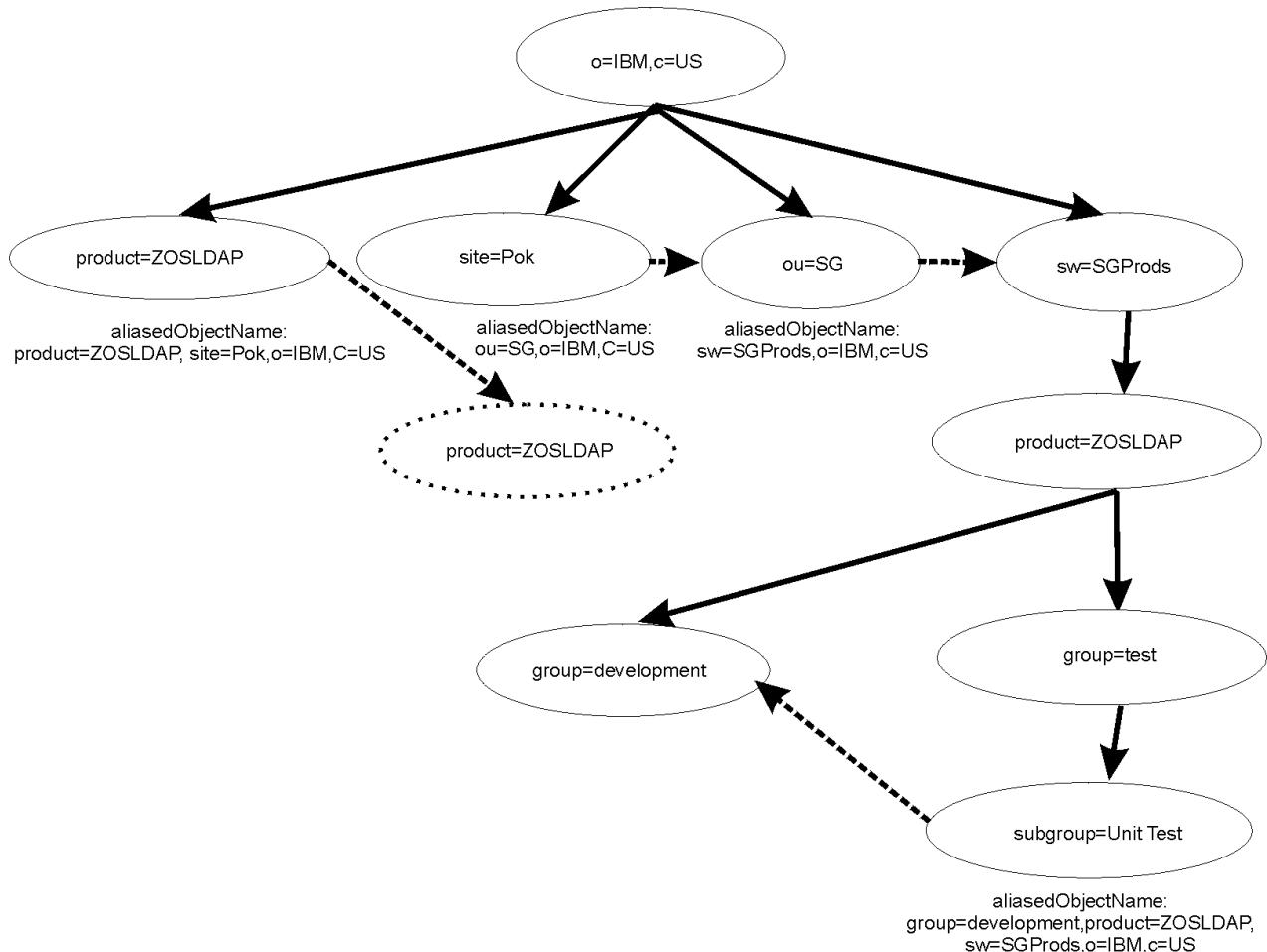
Figure 45. Alias example

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 14. 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 an 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 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 an SDBM or LDBM backend; however, we do not recommend this because that suffix will have to be renamed to avoid an overlap if GDBM is configured in the future.

Change logging is enabled by configuring GDBM in the 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. For more information, see Configuring the LDAP Server in *z/VM: TCP/IP Planning and Customization*. 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 BFS directory.

The **changeLoggingParticipant** configuration option can be used to specify if an LDBM 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 an LDBM, GDBM, or CDBM backend, the default is to create change log entries for changes to that 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

In a GDBM configuration:

1. There can be at most one GDBM backend in the configuration file.
2. The **suffix** option can not 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 file-based GDBM backend

When configuring a file-based GDBM backend, the following configuration file options are required:

```
database GDBM GLDBGD31 [name]
```

The **commitCheckpointEntries, commitCheckpointTOD, databaseDirectory, fileTerminate, filterCacheBypassLimit, filterCacheSize, include, 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. For more information on these options, see Configuring the LDAP Server in *z/VM: TCP/IP Planning and Customization*.

The GDBM database is identical to an LDBM database and is created in the same way.

If you do not want to create change log entries for changes to entries within an LDBM, 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
```

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 an 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 an 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).

LDBM, CDBM, and schema changes

If change logging is activated, each add, modify, delete, or modify DN operation of an entry in any LDBM or CDBM backend or modify of the LDAP server schema entry results in the creation of a change log entry. An exception is 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 LDBM or CDBM backend entry or the LDAP server schema has been committed. This change is not rolled back if the change log record can not 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, and/or resource name 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 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 an LDBM or CDBM modify DN operation

deleteOldRdn
a boolean indicating if the old RDN was deleted in an LDBM or CDBM modify DN operation

newSuperior
the new superior distinguished name specified in an LDBM 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 userid 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 userid 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. The **aclEntry** and **entryOwner** attributes can be entirely deleted, in which case the default ACL is used. For more information, see "Default ACLs with LDBM" on page 153.

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 due to reaching a limit specified by the **changeLogMaxEntries** and **changeLogMaxAge** options in the configuration file. Change log entries can also be deleted by the user through a normal delete operation.

- User operations (search, compare, delete) on change log entries are allowed as long as change logging is enabled (the GDBM backend is configured), even if change logging is off. Add and trim operations by the LDAP server are not performed when change logging is off.

- 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 facilities.

- You can use any attribute in the search filter. A common search is with a "changenumber >= *nnn*" filter, where *nnn* is the largest changenumber value that was retrieved the previous time the search was done (the changenumber=*nnn* entry is retrieved again to ensure that the next part of the change log has not been trimmed).
- The change log entries matching the search filter are returned in increasing changenumber order.
- You cannot depend on there being change log entries for all consecutive change numbers. Some change numbers might be skipped.
- The change log (including the root entry) can be searched as long as change logging is enabled (the GDBM backend is configured), even if change logging is off.

Passwords in change log entries

To avoid including passwords in the **changes** attribute of a change log entry, the 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 5, "Accessing RACF information," on page 61 for more information.

Unloading and loading the change log

DS2LDIF (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. Add operations 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.

Entries are removed until the number of entries remaining is at the limit.

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.

How to set up and use the LDAP server for logging changes

1. Update the LDAP server configuration file:

   a. Add the GDBM backend section, including a change log size and age limit if desired. The following example starts change logging with a maximum size of 1000 entries. Entries are automatically deleted when they become a day old.

   ```
   database gdbm GLDBGD31
   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 needed only when logging changes to resource profiles. Following is an example:

   ```
   database sdbm GLDBSD31
   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 an LDBM or CDBM backend or to the LDAP server schema entry, add the following option to the LDBM, GDBM, or CDBMbackend section (the GDBM backend controls change logging for the schema entry):

   ```
   changeLoggingParticipant off
   ```

2. 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 *z/VM: RACF Security Server Security Administrator's Guide*

3. Restart the LDAP directory server. You will see the GDBM configuration options are displayed.

   For a file-based GDBM backend, 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
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.
```

4. At this point, change logging is started. Depending on your configuration, a change to a RACF user, group, connection, or resource profile, or to an 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.

5. If desired, 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:

```
ldapmdfy  -h ldaphost -p ldapport  -D adminDn  -w adminPw   -f cl.ldif
```

6. 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:

```
ldapsrch -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
```

7. 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, 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 269 for information on retrieving passwords.

8. 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:

```
ldapsrch -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 15. 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 an LDBM backend and how to configure a default referral for the LDAP server.

A referral entry can be added to an 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 might 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 discusses how to associate multiple servers using referrals, an example of associating a set of servers through referrals, basic replication (see Chapter 11, "Basic replication," on page 177), and advanced replication (see Chapter 12, "Advanced replication," on page 193).

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.
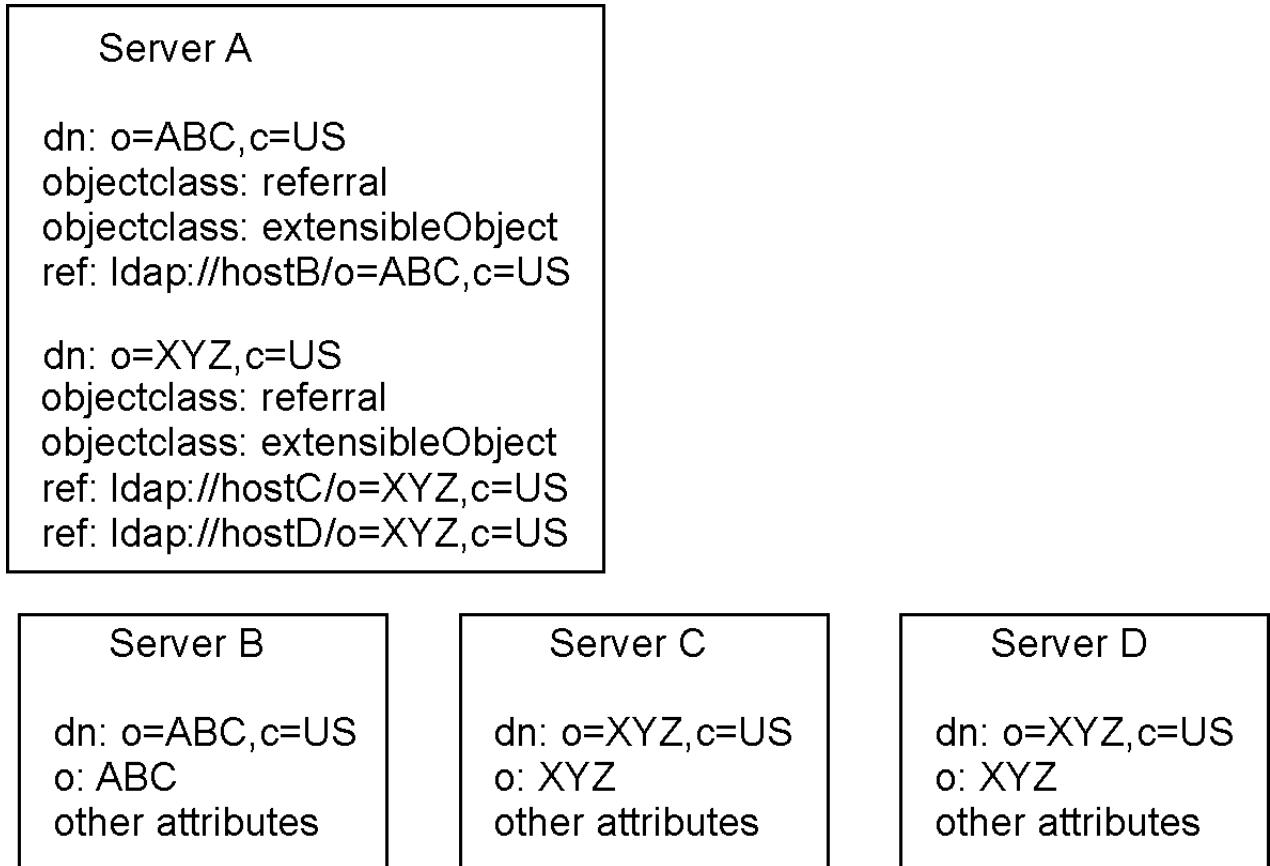
Figure 46. Example using ref attribute

In the example, Server A holds references to two entries: o=ABC,c=US and o=XYZ,c=US. For the o=ABC,c=US entry, Server A holds a reference to Server B and for the o=XYZ,c=US entry, Server A holds references to two equivalent servers, Server C and Server D.

The recommended setup of referrals is to structure the servers into a hierarchy based on the subtrees they manage. Then, provide "forward" referrals from servers that hold higher information and set the default referral to point back to its parent server.

Associating servers with referrals

In order to associate servers through referrals:

- Use referral 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 which are portions of the namespace below this server which are not serviced directly.

Referral entries are created in 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/VM LDAP client can only follow a **ref** value which is in LDAP URL format. A 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/VM 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
```

An 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 that 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. By default, z/VM client utilities chase referrals returned from servers.

Servers present the referral URLs differently depending on the LDAP protocol version being used by the client. Referrals are presented to LDAP 2 clients in the error string, as the protocol does not provide a specific mechanism for indicating referrals. In LDAP 3, protocol elements are specifically defined to allow servers to present referral information to clients.

Using LDAPreferrals

Referrals are not supported by LDAP 2. In order to provide referral information to LDAP 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 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 2. 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 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 3 referrals

In LDAP 3, referrals are defined as part of the protocol. The LDAP 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 LDAP 3: 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 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 3, this difficulty is overcome with a protocol extension in the form of the `manageDsaIT` control. (Appendix B, "Supported server controls," on page 345 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/VM LDAP client utilities support the `-M` option to indicate that the requester is managing the namespace, and therefore wishes to examine and manipulate referral entries as if they were normal entries. For more information, see *z/VM: TCP/IP User's Guide*. 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 349 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 LDAP client utilities such as LDAPSRCH (**ldapsearch**) and LDAPMDFY (**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 chase referrals across servers automatically, 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 an 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.
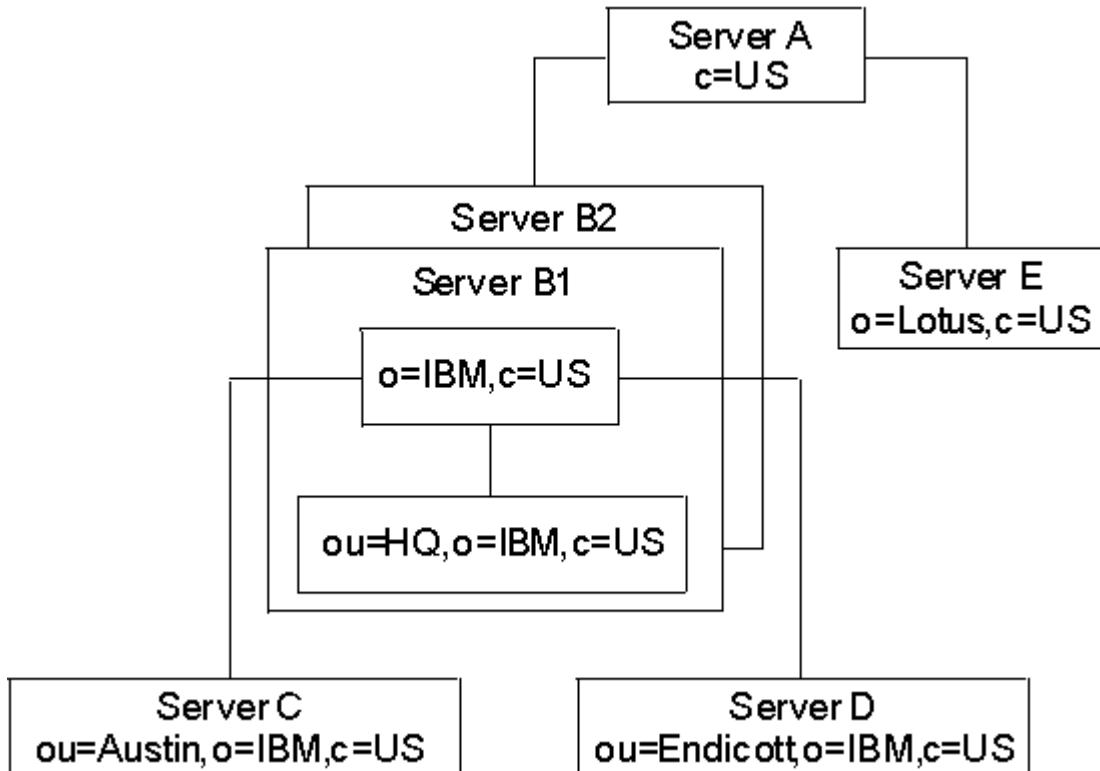
Figure 47. Setting up the servers

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 descendents in other servers.

```
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
```

← Pointer to Servers B1 and B2

← Pointer to Server E

Figure 48. 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
# ldbm database definitions
database ldbm GLDBLD31
suffix "ou=Endicott,o=IBM,c=US"
```

Figure 49. Server D configuration file

5. Putting it all together.

Figure 50 on page 280, Figure 51 on page 281, and Figure 53 on page 283 show these same six servers, showing the referral entries in the database as well as the default referrals which are used for superior references. Also included in Servers B1 and B2 are sample definitions for replication, setting up Server B2 as a replica of Server B1. This ensures that these two servers remains identical. Servers B1 and B2 are located 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 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
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 ldbm GLDBLD31
suffix "o=Lotus,c=US"
```

Directory

```
dn: o=Lotus,c=US
objectClass: organization
```

Figure 50. Referral example summary (servers A and E)

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

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
```

Figure 51. Referral example summary (server B1)

Server B2: Services "o=IBM,c=US"

host name "ibm.com"

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
```

Figure 52. Referral example summary (server B2)

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
```

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 ldbm GLDBLD31
suffix "ou=Endicott,o=IBM,c=US"
```

Directory

```
dn: ou=Directory Team,ou=Endicott,o=IBM,c=US
objectClass: organizationalUnit
```

Figure 53. Referral example summary (servers C and D)

Chapter 16. Organizing the directory namespace

Directory services are meant to help organize the computing environment of the enterprise. To do this, directory services are meant to be used to help find all the resources at one's disposal. Information that is typically found in a directory consists of configuration information for services offered in the enterprise, locating information for people, places, and things in the enterprise, as well as descriptive information about services and resources available in the enterprise. The directory service should be thought of as the spot that can be queried to find whatever is desired in the enterprise.

When designing the format and organization of the directory service for an enterprise, the intended usage scenarios should be considered. These usage characteristics can have an impact on how the directory namespace should be organized so as to offer reasonable performance.

There are two general areas of directory namespace design to be considered. First, the types of information and the layout of where that information will be placed in the directory namespace must be determined. Additional information types can be added at a later date, but there should be some overall design of where in the directory namespace these types of information should be placed. Second, based on the usage characteristics of the users in the enterprise, the number of distinct directory servers and the namespace subtree or subtrees that they support must be considered.

As an example, consider an enterprise that consisted of two physical locations, one in Los Angeles, CA and one in New York City, NY. People in New York City access information about people, places, and things in Los Angeles often, while the people in Los Angeles rarely access information items in New York City. To offer good performance for both locations, a separate directory server could be installed and run in each location. These LDAP servers would manage information about the people, places, and things that reside in their respective locations. In addition, because the New York City personnel access information about things in the Los Angeles location, the information from the Los Angeles LDAP server could be replicated to an additional LDAP server at the New York City LDAP server. This would allow the New York City personnel to access information about the Los Angeles location by contacting a local server. In Los Angeles, however, directory requests about items in the New York City portion of the enterprise namespace are redirected (that is, referred) to the New York City LDAP server for the information. This would save managing a replicated set of information at the expense of slightly longer access times on the less-requested information.

The next two sections discuss information layout in the namespace and partitioning an enterprise namespace across multiple LDAP servers. These sections are followed by a small example.

Information layout

A directory is meant to provide information about people, places, and things in the enterprise. The most direct use of a directory is to hold information on how to contact other people in the enterprise. This has commonly been known as the *internal phone book*. With the widespread enhancements in technology, people are now more accessible than ever. We have pagers, answering machines, cellular phones, and e-mail. In trying to communicate with someone we might need to know about all of this information. Modeling a person object class based on the attributes about a person that are important to others in the enterprise is an easy way to support an online *internal phone book* using an LDAP directory service. In addition to people, different organizations within an enterprise can also be modeled by creating new object classes and attribute types. This would allow storage in the LDAP directory of locating information for useful services in the organization like benefits, travel reservations, and human resources.

Another application of directory services is the ability to model or store information about places. A place could be a conference room, which might have attributes of `numberOfSeats`, `projectorType`, `phoneNumber`, `calendarLocation`, `dataPortType`, `officeNumber`, and `buildingNumber`. Using this method, different conference rooms within a company could be located and compared. Another example of a place would be the whole site in which employees work. An object class for a site LDAP directory entry might be made up of `streetAddress`, `generalManagerDN`, `siteMap`, and `cafeteriaLocation`.

Things abound within the enterprise. Under this category falls computers, copiers, FAX machines, printers, and computer software, as well as configuration information for servers that use an LDAP directory service. Each of these can be modeled with attribute types used inside object classes specific to the device or program.

In laying out where entries should appear in the directory hierarchy, by far the most common method of naming things is to start with the country in which the company is organized, followed by the name of the company, treated as an organization attribute type. Thus, the top level suffix for LDAP directory service names for entries within the company sometimes follows the form: o=CompanyName, c=US (for US-based companies). Alternately, the top level suffix may follow the domain form, for example: dc=CompanyName,dc=com. Below this suffix it is common for organizational unit object classes to be used to represent departments or sites within an organization. Below these organizational entries the actual entry representing a person, place, or thing would be defined. When organizing the information layout for the namespace, the intended usage should be considered to ensure the best performance.

Example of building an enterprise directory namespace

Let us look at an example configuration that exhibits the features available with the LDAP server. To set the stage, we will consider a moderately sized company that has personnel working in three locations across the United States. Big Company, Inc. has corporate headquarters in Chicago, IL, and two satellite facilities, one in Los Angeles, CA and the other in New York City, NY. The information technology staff would like to make available information about all of the company's computing and office services using an LDAP directory. In order to facilitate local modifications as necessary of the information in the directory, as well as provide improved response time for accessing local information, each site will have an LDAP server running. The server running at each site will be responsible for managing the directory information that pertains to that site.

The first thing to do is determine the name of the root of the directory namespace for Big Company, Inc. Typically, the name for the company will consist of the country of origin along with the company's given name. In LDAP directory terminology, the company is an organization. In this example, we chose:

```
o=Big Company, c=US
```

as the company's name is Big Company and is located in the United States. Choosing a name of this format helps ensure that when a global namespace coordinator is established, the company's chosen *root* will fit nicely into the overall directory namespace.

Next to choose are the names of the three locations under which the directory information is stored. At this point, the namespace could be organized in a number of ways. One way would be to organize by functional unit (regardless of location). This model is useful if individuals (or computers, or other equipment or services) typically remain with the functional unit as opposed to being tied to the individual or physical location. Another way would be to organize based on the physical locations of the parts of the organization. This is useful if the people, places, and things to be stored in the directory typically do not move between locations. This latter approach will be used in the example. So, with three locations, three names are defined below the overall company distinguished name:

```
ou=Los Angeles, o=Big Company, c=US
ou=Chicago, o=Big Company, c=US
ou=New York City, o=Big Company, c=US
```

Since separate LDAP servers will be established at each location, these names represent the root of the subtree stored and managed by the directory server at each location.

For administration, each site will have a different directory administrator. To define this administrator, an administrator distinguished name and password need to be defined for each location. To start, the following names will be used:

```
AdminDN "cn=Administrator, ou=Los Angeles, o=Big Company, c=US"

AdminDN "cn=Administrator, ou=Chicago, o=Big Company, c=US"
```

```
AdminDN "cn=Administrator, ou=New York City, o=Big Company, c=US"
```

Since the Chicago location is also the corporate headquarters, the LDAP directory at this location will be used to store information about the entire company as well as information about the Chicago site.

We now have enough information to set up the base configuration files for each of the three LDAP servers that will be used to supply this information. Following are the files needed to set up the LDAP servers on each site. Note that what is shown is the minimal setup required. Other options could be specified in addition to these. For configuration options, see <u>Creating the DS CONF File</u> in *z/VM: TCP/IP Planning and Customization*.

```
# Configuration file for the Chicago LDAP server
adminDN "cn=Administrator, ou=Chicago, o=Big Company, c=US"

database ldbm GLDBLD31
suffix "o=Big Company, c=US"
# end of configuration file
```

Figure 54. Chicago base configuration

```
# Configuration file for the Los Angeles LDAP server
referral ldap://ldap.chicago.bigcompany.com
adminDN "cn=Administrator, ou=Los Angeles, o=Big Company, c=US"

database ldbm GLDBLD31
suffix "ou=Los Angeles, o=Big Company, c=US"
# end of configuration file
```

Figure 55. Los Angeles base configuration

```
# Configuration file for the New York City LDAP server
referral ldap://ldap.chicago.bigcompany.com

adminDN "cn=Administrator, ou=New York City, o=Big Company, c=US"

database ldbm GLDBLD31
suffix "ou=New York City, o=Big Company, c=US"
# end of configuration file
```

Figure 56. New York City base configuration

The referral line indicates the default place to refer connecting clients when the LDAP server does not contain the information requested by the client. It is called the *default referral*. It is in the form of an LDAP URL. After the scheme name (ldap), the LDAP URL contains a TCP/IP DNS host name for another LDAP server. In this example, it is assumed that the TCP/IP host on which the Chicago LDAP server is running is ldap.chicago.bigcompany.com. The Chicago LDAP server does not have a default referral defined. This keeps directory searches from inadvertently going over the Internet from within the company.

The adminDN line indicates the distinguished name that should be used to connect to the LDAP server in order to have complete control over the data content held by the LDAP server.

The database line indicates that all following lines pertain to the LDBM storage method. The suffix line indicates what part of the namespace is contained in this server.

After these files have been created, one or more of the LDAP servers can be started. However, there will be no initial data in the LDBM database. The next section tells you how to load entries into the LDAP server.

Priming the directory servers with information

Add entries to an LDBM (file-based) backend in the LDAP server by using LDAPADD and LDAPMDFY (the **ldapadd** and **ldapmodify** utilities) or by using the LDAP C language API and the LDAP protocol. It is recommended that at least the top levels of directory information be loaded first into the database. This provides a base from which to add more entries into the directory namespace.

Using LDIF format to represent LDAP entries

The LDAP Data Interchange Format (LDIF) is used to represent LDAP entries in a simple text format. An LDIF file contains groups of attribute information which will be treated as an entry to be added to the directory. The general format of an LDIF entry is:

```
dn: distinguished name
attrtype1: attrvalue1
attrtype2: attrvalue2
…
```

Each line in the LDIF file must begin in column 1. However, to continue a line, start the next line with a single space or tab character. For example:

```
dn: ou=departments, ou=New York City, o=Big Co
 mpany, c=US
```

Multiple attribute values are specified on separate lines. For example:

```
objectclass: organizationalunit
ou: departments
```

> **Note about editing LDIF files** Be aware that some editors place blank spaces at ends of all empty lines within a file. A blank space at the beginning of a line signifies continuation of the entry. The blank lines used to separate entries may be treated as continuations of an attribute value instead of separators if an editor has modified the LDIF file. Also, be aware that some editors delete blanks at the end of a line that is not empty. This can change the value of an attribute, especially if that value is continued on the next line.

If an *attrvalue* contains a nonprinting character, or begins with a space or a colon (:), the *attrtype* is followed by a double colon (::) and the value is encoded in base64 notation. For example, the value:

```
" begins with a space"
```

would be encoded like this:

```
cn:: IGJlZ2lucyB3aXRoIGEgc3BhY2U=
```

Multiple entries within the same LDIF file are separated by blank lines. Here is an example of an LDIF file containing three entries.

```
dn: ou=New York City, o=Big Company, c=US
objectclass: organizationalunit
ou: New York City

dn: ou=fax machines, ou=New York City, o=Big Company, c=US
objectclass: organizationalunit
ou: fax machines

dn: ou=computers, ou=New York City, o=Big Company, c=US
objectclass: organizationalunit
ou: computers
```

Note: Trailing spaces are not trimmed from values in an LDIF file. Also, multiple internal spaces are not compressed. If you do not want them in your data, do not put them there.

Multiple attribute values for the same attribute type are specified on multiple lines within the specification of a directory entry. For example:

```
dn: cn=John Doe, ou=New York City, o=Big Company, c=US
objectclass: person
cn: John Doe
phonenumber: 555-1111
phonenumber: 555-2222
sn: Doe
```

Generating the file

A file is typically generated using an existing source of information and some tools to format the data into the LDIF format. Note that the order of entries in the LDIF file is important. In order for an entry specified in the LDIF file to be successfully added to the directory, its parent entry must first exist in the directory namespace. For this reason, the top level entries in the directory namespace subtree that the particular LDAP server will support must be first in the LDIF file.

For our example, we will define just a minimal set of entries to get the directory server useful at each location. This will include two referral entries for the Chicago location. The meaning of these entries will be discussed in more detail in the following sections.

Here is the base set of LDIF files to set up the directory namespace at each location. For the Los Angeles location:

```
dn: ou=Los Angeles, o=Big Company, c=US
objectclass: organizationalunit
ou: Los Angeles

dn: cn=Administrator, ou=LosAngeles, o=Big Company, c=US
objectclass: person
cn: Administrator
sn: Administrator
userpassword: xxxxx

dn: ou=fax machines, ou=Los Angeles, o=Big Company, c=US
objectclass: organizationalunit
ou: fax machines

dn: ou=computers, ou=Los Angeles, o=Big Company, c=US
objectclass: organizationalunit
ou: computers

dn: ou=departments, ou=Los Angeles, o=Big Company, c=US
objectclass: organizationalunit
ou: departments
```

For the New York City location:

```
dn: ou=New York City, o=Big Company, c=US
objectclass: organizationalunit
ou: New York City

dn: cn=Administrator, ou=New York City, o=Big Company, c=US
objectclass: person
cn: Administrator
sn: Administrator
userpassword: xxxxx

dn: ou=fax machines, ou=New York City, o=Big Company, c=US
objectclass: organizationalunit
ou: fax machines

dn: ou=computers, ou=New York City, o=Big Company, c=US
objectclass: organizationalunit
ou: computers

dn: ou=departments, ou=New York City, o=Big Company, c=US
```

```
objectclass: organizationalunit
ou: departments
```

For the Chicago location:

```
dn: o=Big Company, c=US
objectclass: organization
o: Big Company

dn: ou=Los Angeles, o=Big Company, c=US
objectclass: referral
objectclass: extensibleObject
ref: ldap://ldap.losangeles.bigcompany.com/ou=Los Angeles,o=Big Company,c=US

dn: ou=New York City, o=Big Company, c=US
objectclass: referral
objectclass: extensibleObject
ref: ldap://ldap.newyorkcity.bigcompany.com/ou=New York City,o=Big Company,c=US

dn: ou=Chicago, o=Big Company, c=US
objectclass: organizationalunit
ou: Chicago

dn: cn=Administrator, ou=Chicago, o=Big Company, c=US
objectclass: person
cn: Administrator
sn: Administrator
userpassword: xxxxx

dn: ou=fax machines, ou=Chicago, o=Big Company, c=US
objectclass: organizationalunit
ou: fax machines

dn: ou=computers, ou=Chicago, o=Big Company, c=US
objectclass: organizationalunit
ou: computers

dn: ou=departments, ou=Chicago, o=Big Company, c=US
objectclass: organizationalunit
ou: departments
```

These files will now be used with a load facility. After loading these files on each respective directory server system, the directory namespace will be formed and the servers can now be used to hold and supply information.

Two entries added to the Chicago location directory server database deserve some special attention. These are the referral objects that were in the LDIF file for the Chicago location. Notice that the referral objects have the identical distinguished name as the root of the LDAP directory namespace that is served by the Los Angeles and New York City servers. These entries, coupled with the default referral specification in the configuration file for the directory servers in Los Angeles and New York City, enable searches of the Big Company namespace to originate at any of the three directory servers and resolve to the correct server to obtain the information.

A referral redirects a client request to a different LDAP server that can presumably handle the request (or refer the client to another server that can handle the request). In our example, if a client connects to the New York City server requesting a name that is under the Los Angeles portion of the namespace, the New York City server will send back a referral to the client based on the default referral. This will point the client at the Chicago directory server. The Chicago server will resolve the request down to the referral object for distinguished name ou=Los Angeles, o=Big Company, c=US and refer the client to the Los Angeles server. Finally, the client will contact the Los Angeles server and obtain the information requested.

Setting up for replication

As people start using the directory service in their daily routines at Big Company, Inc., the information technology staff notices that the people in New York City are doing a lot of work with the people in Los Angeles. So much, in fact, that an analysis of the TCP/IP traffic between New York City and Los Angeles shows that much of the traffic is directory access requests, presumably to look up phone numbers or FAX numbers for people in Los Angeles. The information technology staff decides to improve directory lookup

response time, as well as lessen the directory lookup traffic between New York City and Los Angeles, by creating a replica of the Los Angeles directory server's information in New York City. This will allow local access to this information by the New York City users and cut down on the amount of requests from New York that must travel to Los Angeles to be completed.

Defining another LDAP server

To set up a replica of the LDAP server information in Los Angeles, a second LDAP server must be defined and started in New York City. This server can reside on the same system as the first LDAP server, though if this is chosen, the TCP/IP port that this replica server listens on must be different from the other LDAP server running on the system. As an alternative, the replica server could run on a different system, allowing it to listen on the default LDAP port. The configuration file for the replica server in New York City will be very similar to the configuration files for the New York City server and the Los Angeles server. This configuration file must contain some additional items that pertain to replication. Here is what the contents of the New York City Los Angeles replica server should contain:

```
# Configuration file for the New York City Los Angeles replica LDAP server
referral ldap://ldap.chicago.bigcompany.com

listen ldap://:2001
adminDN "cn=Administrator, ou=Los Angeles, o=Big Company, c=US"

database ldbm GLDBLD31
suffix "ou=Los Angeles, o=Big Company, c=US"

masterServer ldap://ldap.losangeles.bigcompany.com
masterServerDN "cn=Replicator, ou=Los Angeles, o=Big Company, c=US"
# end of configuration file
```

The additional lines at the end of the configuration file specify the only "user" that can update entries in the replica LDAP server. The values here must match the values entered at the "source" location when the replica is defined.

Preparing the replica

The next step is to get the LDAP replica primed with the existing information in the Los Angeles server and set up the Los Angeles server to replicate to the New York City replica. The set of steps to perform (described in "Populating a replica" on page 183) ensures that the replicas are in sync and that no update is lost during this synchronization. Once the replica is defined at the source location, updates to the directory information will be logged to be sent to the replica server when possible.

To initially synchronize the data between the LDAP master server and the LDAP replica server, perform the steps in "Populating a replica" on page 183.

While there are a number of manual steps to perform, there is a small consolation that the steps at different locations are not interleaved. All work can be done at the source location and then all work can be performed at the target (replica) location.

Resynching the replica and master servers

If it is noticed that a replica's contents are out of sync with the information at the master server, the information can be resynched by following the steps shown in "Recovering from basic replication out-of-sync conditions " on page 190.

Notifying users of the replica

At this point, the New York City users can be notified that a second LDAP server is now available for their use. The notification should contain either the LDAP URL of the new LDAP replica server or the host name and port number of the LDAP replica server, as well as the base of the LDAP subtree that is published by the replica. As updates are made to the Los Angeles LDAP server, these updates will be propagated to

the replica server in New York City. See Chapter 11, "Basic replication," on page 177 for more details on replication.

What Big Company, Inc. now has in place is an Enterprise Directory service that can be used by whatever enterprise distributed processing tasks require lookup or configuration information. These enterprise distributed processing tasks and applications may require some changes to make use of the directory service, but the result will be the ability to view, find, and modify the configuration of the enterprise by looking at and modifying the contents of the LDAP directory.

Chapter 17. Client considerations

When an LDAP application is communicating with an LDAP server, you should consider the following special topics:

- Root DSE
- Monitor Support
- CRAM-MD5 authentication support
- UTF-8 data over LDAP 2
- Attribute types stored and retrieved in lowercase
- Abandon behavior
- Changed return codes
- Reason codes.

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 supported is **objectclass=***. There is no access control checking for the root DSE, but an anonymous bind will fail 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 configured in the LDAP server.

The following example uses LDAPSRCH (the **ldapsearch** utility) to request a base search of the root DSE and shows sample output for the search:

```
ldapsrch -h ldaphost -p ldapport -s base -b "" "objectclass=*"
```

Following is an example of the information that the LDAP server will report on a search of the root DSE. A subset of these values may appear in your root DSE based on the server configuration choices you have made.

```
vendorname=International Business Machines (IBM)
vendorversion=z/VM V6R4
ibmdirectoryversion=z/VM V6R4
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 50. 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 which 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 is capable of retaining changelog entries based on age.
1.3.18.0.2.32.24	Monitor operation counts	The server provides new monitor operation counts for initiated and completed operation types.

Table 50. 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 or CDBM entries in the server.
1.3.18.0.2.32.28	TLS capabilities	Specifies that the server is capable of performing Transport Layer Security (TLS).
1.3.18.0.2.32.29	Non-blocking advanced replication	Indicates that this server is capable of ignoring some errors received from a consumer (replica) server that would normally cause an update to be retransmitted periodically until a successful return code is received.
1.3.18.0.2.32.31	**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 nested 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 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 will also be 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.

OID assigned	Short name	Description
		Table 50. Object Identifiers (OIDs) for supported and enabled capabilities (continued)
1.3.18.0.2.32.51	Advanced replication conflict resolution maximum entry size	Indicates that this server supports the **ibm-slapdReplConflictMaxEntrySize** attribute on an CDBM entry with an objectclass of **ibm-slapdReplicationConfiguration**. This attribute value indicates the maximum number of bytes that an entry can contain and still 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 which 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 max consecutive repeated characters	Indicates that this server supports password policies that restrict the maximum number of consecutive repeated characters in password values.

Table 50. Object Identifiers (OIDs) for supported and enabled capabilities (continued)		
OID assigned	Short name	Description
1.3.18.0.2.32.94	Fine grained timestamps	Indicates that this server supports advanced replication with fine grained timestamps 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 and CDBM backends 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 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 and CDBM suffix. These individual searches are each limited by the **timelimit** and **sizelimit** options specified in the LDAP server configuration file or in the requestor'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 left to return when that individual search is started. For more information, see the descriptions of the **sizelimit** and **timelimit** options in Step 6. Create and Customize the LDAP Configuration File (DS CONF) in *z/VM: TCP/IP Planning and Customization*. See Managing group search limits for more information about group search limits. Each individual LDBM and CDBM search is subject to the normal LDBM and CDBM access control checking.

The following example uses LDAPSRCH 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.

```
ldapsrch -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
```

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 307.

CRAM-MD5 authentication support

CRAM-MD5 authentication is supported on the IBM Tivoli Directory Server client utilities on other platforms, such as AIX®, Linux®, and Windows. However, the manner in which it has been implemented on the IBM Tivoli Directory Server on other platforms varies from the support that is available on the LDAP server.

In order to perform a CRAM-MD5 authentication bind with the IBM Tivoli Directory Server client utilities on other platforms to the LDAP server, you must specify the bindDN with the **-D** option. The IBM Tivoli Directory Server client utilities on other platforms do not support the specification of the username on a CRAM-MD5 bind.

UTF-8 data over LDAP 2

LDAP 3 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 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 LDAP 2.

Note: Different clients treat non-IA5 data differently over LDAP 2. Refer to the documentation for the client APIs you are using for more information.

Attribute types stored and returned in lowercase

The LDAP server stores and returns attribute types in lowercase (normalized). For example, the attribute type "`productName`" is returned as "`productname`".

Abandon behavior

The LDAP server reads additional operations as they arrive as long as the connection is not a secure connection and the previous operation is not bind, unbind, or extended operation. This allows the LDAP server to process abandon operations as they are received and affect previously submitted operations.

Chapter 18. SSL Certificate/Key Management and SSL Tracing Information

SSL connections make use of public/private key mechanisms for authenticating each side of the SSL session and agreeing on bulk encryption keys to be used for the SSL session. To use *public/private key mechanisms*, or *PKI*, public/private key pairs must be generated. In addition, X.509 certificates (which contain public keys) might need to be created, or certificates must be requested, received, and managed.

SSL uses the gskkyman utility to manage PKI private keys and certificates. To call the gskkyman utility, issue the GSKKYMAN command. Note that GSKKYMAN operates in a ZCMS environment only. The gskkyman utility creates, fills in, and manages a file that contains PKI private keys, certificate requests, and certificates. This file is called a *key database*. By convention, it has a file name extension of .kdb.

SSL uses the GSK_KEYRING_FILE environment variable to specify the locations of the PKI private keys and certificates. The key database file name is passed in this environment variable.

Key Database Files

Key database files are password protected because they contain the private keys that are associated with some of the certificates that are contained in the key database. Private keys, as their name implies, should be protected because their value is used in verifying the authenticity of requests made during PKI operations.

It is recommended that key database files be set with the following string of file permissions:

```
rw- --- ---  (600) (read-write for only the owner of the key database)
```

The owner of the key database should be the user who manages the key database. The user ID that runs the LDAP server must have at least read permission to the key database file at runtime. If the LDAP server user ID is a server program that runs under a different user ID than the administrator of the key database file, it is recommended that a group be setup to control access to the key database file. In this case, it is recommended that you set the permissions on the key database file to the following:

```
rw- r-- ---  (640) (read-write for owner and read-only for group)
```

The owner of the key database file is set to the administrator user ID and the group owner of the key database file is set to the group that contains the server that will be using the key database file.

For more information about gskkyman and setting up the key database and its permissions, see SSL Certificate Management in *z/VM: TCP/IP User's Guide*.

SSL Tracing Information

SSL tracing techniques are for use primarily by IBM service personnel in determining the cause of an SSL problem. If you encounter a problem and call the IBM Support Center, you may be asked to obtain trace information or enable one or more of the diagnostic messages described below.

Use the gsktrace utility to create a readable copy of SSL trace information. For information about gsktrace, see SSL Tracing in *z/VM: TCP/IP User's Guide*.

gsktrace is not intended for use in a production environment and is used for diagnostic purposes only.

Chapter 19. Performance tuning

Several server configuration options and facilities significantly affect the performance of the server. In addition, specific LDAP server backends operate in conjunction with other products that may require tuning to accommodate the LDAP server. For example, 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, and other system resources associated with threads. Therefore, you may want to avoid making this option larger than is needed.

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

Debug settings

Activating the LDAP server debug trace facility impacts performance. If optimal performance is desired, debug should only be activated when it is necessary to capture diagnostic information.

Storage in the LDAP virtual machine

The LDAP server generally requires a minimum of 96 megabytes to run. 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 may increase the amount of storage required by the server.

Some caches are invalidated by update activities. If this is a frequent occurrence, increasing the cache size may be of little or no benefit. If the cache hit rate is never any higher than zero for a particular cache, the cache can be disabled by setting its size to 0. However, even caches with seemingly low cache hit rates might provide some benefit, therefore, you should generally avoid disabling them unless close monitoring is done to ensure 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 SMSG *ldapsrv* DISPLAY MONITOR 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, since the caches are backend specific.

- Examine the cache hit rate, the current number of entries, and the maximum allowed entries (configured size). Also, note the number of cache refreshes and the average size of the cache at refresh.

- If the cache hit rate is well below 100% and the cache is frequently fully populated, consider increasing the cache size. Since this is a configuration option, you must change the server configuration file and restart the server to affect the change.

The following caches are implemented in the LDAP server:

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.

filter cache
This cache holds information related to the mapping of search request inputs and the result set. This cache is implemented in the LDBM, CDBM, and GDBM backends. For LDBM, CDBM, and file-based GDBM, retrieval of information from this cache avoids database read operations when processing search requests.

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 307 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 54 on page 311 for more information about the **cn=operations,cn=monitor** entry and its attributes.

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**

For more details about these attributes, see "Password policy operational attributes" on page 116.

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 a user's effective password policy, 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 user's group membership must be resolved to properly apply group password policies.

LDBM performance considerations

The LDAP server LDBM backend uses the BFS file system for its persistent storage of the directory entry data. When the LDAP server is executing, the entire directory contents are held in virtual storage, including index structures for quick access.

Holding the entire directory contents in virtual storage 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 occur only within the LDAP server virtual storage, and are not stored on DASD.

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 virtual machine
- 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 virtual machine for LDBM data

Since the entire LDBM directory is kept in storage in the LDAP virtual machine, you need to plan accordingly. The amount of storage required can be estimated from the size of the LDIF data used to load the directory. The storage needed to contain the data is about 7 to 10 times the size of the LDIF file.

These are estimates only. Furthermore, these estimates pertain only to the storage required to hold the LDBM directory representation. You must plan for additional storage for running the server as mentioned in "Storage in the LDAP virtual machine" on page 303.

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.

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 file.
- 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 may 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 BFS 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.

CDBM performance considerations

If the CDBM backend is used only to store configuration entries, then no tuning is necessary. However, if the CDBM backend is used to store user defined entries, for more information about CDBM tuning, see "LDBM performance considerations" on page 305.

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 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 SMSG command. The command is:

```
smsg server_id display monitor
```

where *server_id* is the LDAP virtual machine user ID.

Statistics generally reflect data gathered since the LDAP server was started. However, many of the counters can be reset by using the SMSG command. The command is:

```
smsg server_id reset monitor
```

where *server_id* is the LDAP virtual machine user ID. In this case, the values reflect data gathered since the last reset.

The monitor search returns the following attributes:

Table 51. Server statistics

Attribute	Meaning
currentconnections	Current number of client connections
currenttime	Current date and time on the server

Table 51. Server statistics (continued)

Attribute	Meaning
livethreads	Configured number of communication threads (**commThreads**)
maxconnections	Configured maximum number of 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
timedoutpagesets	Number of paged search result sets abandoned based on the **idleConnectionTimeout** setting
totalconnections	Number of client connections 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 may be lower than the **maxconnections** value because of system limits. If the value for the **maxConnections** configuration option is not valid, the **maxconnections** attribute value on **cn=monitor** search 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 in *z/VM: TCP/IP Planning and Customization*.

When statistics are reset, **resettime** is set to the value of **currenttime, resets** is incremented, and **maxreachedconnections** is set to the value of **currentconnections**. None of the other server statistics listed above are affected by a reset.

Table 52. Server and backend specific statistics

Attribute	Meaning
abandonsrequested	Number of abandon operations requested

Table 52. Server and backend specific statistics (continued)

Attribute	Meaning
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
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
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 53. Backend specific statistics

Attribute	Meaning
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

Table 53. Backend specific statistics (continued)

Attribute	Meaning
acl_source_cache_hit	Number of lookups that have hit the ACL Source cache
acl_source_cache_miss	Number of lookups that have missed the ACL Source cache
acl_source_cache_percent_hit	Percent of lookups that have hit the ACL Source cache
acl_source_cache_refresh	Number of times the ACL Source cache was invalidated
acl_source_cache_refresh_avgsize	Average number of entries in the ACL Source cache at invalidation
dn_cache_size	Configured maximum size (in entries) of the DN cache (**dnCacheSize**)
dn_cache_current	Current size (in entries) of the DN cache
dn_cache_hit	Number of lookups that have hit the DN cache
dn_cache_miss	Number of lookups that have missed the DN cache
dn_cache_percent_hit	Percent of lookups that have hit the DN cache
dn_cache_refresh	Number of times the DN cache was invalidated
dn_cache_refresh_avgsize	Average number of entries in the DN cache at invalidation
dn_to_eid_cache_size	Configured maximum size (in entries) of the DN to Entry ID cache (**dnToEidCacheSize**)
dn_to_eid_cache_current	Current size (in entries) of the DN to Entry ID cache
dn_to_eid_cache_hit	Number of lookups that have hit the DN to Entry ID cache
dn_to_eid_cache_miss	Number of lookups that have missed the DN to Entry ID cache
dn_to_eid_cache_percent_hit	Percent of lookups that have hit the DN to Entry ID cache
dn_to_eid_cache_refresh	Number of times the DN to Entry ID cache was invalidated
dn_to_eid_cache_refresh_avgsize	Average number of entries in the DN to Entry ID cache at invalidation
entry_cache_size	Configured maximum size (in entries) of the Entry cache (**entryCacheSize**)
entry_cache_current	Current size (in entries) of the Entry cache
entry_cache_hit	Number of lookups that have hit the Entry cache
entry_cache_miss	Number of lookups that have missed the Entry cache
entry_cache_percent_hit	Percent of lookups that have hit the Entry cache
entry_cache_refresh	Number of times the Entry cache was invalidated
entry_cache_refresh_avgsize	Average number of entries in the Entry cache at invalidation
entry_owner_cache_size	Configured maximum size (in entries) of the Entry Owner cache (**entryOwnerCacheSize**)

Table 53. Backend specific statistics (continued)

Attribute	Meaning
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 backend report **filter_cache** statistics. A file-based GDBM backend 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 54. Operations monitor statistics

Attribute	Meaning
cachesize	Configured maximum number of search patterns in the operations monitor (**operationsMonitorSize**)
currenttimestamp	Current date and time in Zulu time stamp format
entries	Total number of search patterns in the operations monitor entry

Table 54. Operations monitor statistics (continued)

Attribute	Meaning
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 Zulu time stamp format of last reset or server start up 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 Zulu 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 Zulu 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 client's IP address.

The format of the **searchIPStats** and **searchStats** attribute values is:

```
ldap://clientIP/baseDN?attributes?scope?filter-string?status,numpOps=numOps,avg=avg,
rate=rate,maxRate=maxRate,maxRateTimeStamp=maxRateTimeStamp,
createTimeStamp=createTimeStamp
```

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 Zulu 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 Zulu 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 54 on page 311 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 SMSG DISPLAY MONITOR command, see SMSG Interface to the LDAP Server in *z/VM: TCP/IP Planning and Customization*.

Note: DISPLAY MONITOR output does not display **cn=operations,cn=monitor** data.

Monitor search examples

An example of a monitor search using scope=base follows. This returns only statistics related to the entire server:

```
ldapsrch -h ldaphost -p ldapport -b cn=monitor -s base objectclass=*
```

```
cn=monitor
version=z/VM Version 6 Release 4 IBM LDAP Server

livethreads=10
maxconnections=24982
sysmaxconnections=25000
totalconnections=20709
currentconnections=1
maxreachedconnections=15
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
entriessent=4221
bytessent=1564656734
searchreferencessent=0
currenttime=Thu Sep 25 16:33:00.187846 2008
starttime=Thu Sep 25 15:52:21.693392 2008
resettime=Thu Sep 25 15:52:21.693392 2008
resets=0
```

Following is an example of output of a monitor search with scope=one for a server configured with an LDBM backend. This example shows backend-specific statistics and operations monitor statistics. The cache statistics shown would be included only for LDBM, CDBM, GDBM, and schema backends, because 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.

```
ldapsrch -L -h ldaphost -p ldapport -b cn=monitor -s one objectclass=*
```

```
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=20080313132741.415477Z,createTimeStamp=20080313132628.36161
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=20080313132626.
545031Z,createTimeStamp=20080313132615.953823Z,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=20080313132628
.047031Z,createTimeStamp=20080313132626.878552Z,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=20080313132741.415477Z,createTimeStamp=2008031
3132628.361618Z,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=20080313132626.545031Z,createTimeStamp=20080313132615.953823Z,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=200
80313132628.047031Z,createTimeStamp=20080313132626.878552Z,ID=2742
currenttimestamp: 20080313132836.785259Z
resettimestamp: 20080313132615.369362Z
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 9, "Static, dynamic, and nested groups," on page 133 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 317 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" on page 316.

- 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/VM LDAP might experience performance problems and increased storage usage in the LDAP server as 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

Some scenarios that require substantial amounts of processing and storage within the 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 an LDBM backend that contains the large entry.

These scenarios are also susceptible to the affects of LE HEAPPOOL usage as described below.

The addressability limits of the LDAP server may 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:

- Increase the virtual machine size, if possible.
- Limit the number of members 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 defined for the LDAP server, and the level of concurrent activity within the server.

- If possible, avoid configuring **persistentSearch on** for an LDBM backend that contains large entries. Some applications that exploit persistent search may only do so with 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.

LE heap pool considerations

By default, the z/VM LDAP server uses LE heap pools to improve performance. This facility reduces the processor consumption and allows better parallelism of concurrent requests within the 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, once 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 LDAP server must process a large access group entry in storage, the following may occur:

- While the request is processing, the LDAP server may use all available storage in its virtual machine, 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 LDAP server may be allocated and reserved to specific heap pools. Although the LDAP server may appear to be available and able to process a variety of requests, many subsequent requests may 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 LDAP server may be detectable by system resource monitoring products, such as Performance Toolkit for z/VM.

If these problems occur, consider either tuning the heap pool sizes or disabling the heap pools for the LDAP server.

Tuning the heap pool sizes optimizes storage usage for the data within the LDAP server. See z/OS: Language Environment Programming Guide (https://www.ibm.com/docs/en/SSLTBW_3.1.0/pdf/ceea200_v3r1.pdf) for details on how to tune the heap pool settings. Note that the procedure for tuning heap pool settings requires a controlled environment with representative workloads. In this case, the workload should include the scenarios described earlier that cause the large demands for storage. Note that it is recommended that the storage reports needed for the tuning procedure be gathered in a non-production environment because tracking the storage statistics significantly impacts performance.

Disabling heap pools reduces the total heap storage requirements of the LDAP server, at the cost of increased processing.

Overriding the heap pool settings for the LDAP server can be done by specifying the LE runtime option HEAPPOOLS. This option can be specified on the :parms. tag in the DTCPARMS file. For more information about setting this parameter, see z/OS: Language Environment Programming Reference (https://www.ibm.com/docs/en/SSLTBW_3.1.0/pdf/ceea300_v3r1.pdf) and z/VM: Language Environment User's 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.

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 136 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 Configuring the LDAP Server in *z/VM: TCP/IP Planning and Customization* for more information about the **ibm-slapdPagedResLmt** and **ibm-slapdPagedResAllowNonAdmin** attributes in the **cn=configuration** entry. Also, see "pagedResults" on page 347 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.

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 Configuring the LDAP Server in *z/VM: TCP/IP Planning and Customization* for more information about the **ibm-slapdSortKeyLimit** and **ibm-slapdSortSrchAllowNonAdmin** attributes in the **cn=configuration** entry. Also, see "SortKeyRequest" on page 354 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 actually needed.

The following should be noted:

- The distinguished names (DNs) of entries and the searchable attributes within entries in GDBM tend to be well bounded in size and content. As such, the default sizes for the **DN_TRUNC** column in the **DIR_ENTRY** table and the **VALUE** column in the **DIR_SEARCH** table do not require adjustment.

- Since most GDBM requests are update operations, the search filter cache is disabled by default. You may enable the cache, if desired, but if this is done, it is recommended that the cache is monitored to ensure it is providing a benefit.

- 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 in *z/VM: TCP/IP Planning and Customization*.

SDBM performance considerations

The z/VM LDAP server SDBM backend allows access to the RACF database. Most tuning that affects performance in this area is within the RACF product.

Also, see "SDBM operational behavior" on page 70 for details regarding 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 345 for more information.

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. For more information, see "Updating the schema" on page 32.

```
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 an OS/390 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 OS/390 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 OS/390 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' )
     DESC '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=( 1.3.18.0.2.4.242 NAME ( 'racfAuthKeyword' )
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
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=""> <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\V6.4\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\V6.4\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\V6.4\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=( 1.3.18.0.2.4.3417 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 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2326 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2327 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2328 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2329 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.233 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2330 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2331 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2332 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2333 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2334 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2335 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2336 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2337 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2338 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2339 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.234 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2340 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2341 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2342 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2343 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2344 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2345 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2346 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2347 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2348 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2349 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.235 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2350 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2351 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2352 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2353 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2354 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.2355 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2356 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2357 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2358 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2359 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.236 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2360 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2361 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2367 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.237 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2376 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.2377 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.238 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2380 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.2381 ACCESS-CLASS critical )
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 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.2409 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.241 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2411 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.242 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2420 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.2425 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.2428 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.243 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2430 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2433 ACCESS-CLASS normal )
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 )
```

Initial LDAP server schema

```
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 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2486 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.249 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2493 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2494 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2495 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2496 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2497 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.2498 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.250 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2500 ACCESS-CLASS critical )
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 ACCESS-CLASS critical )
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 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.304 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3072 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3073 ACCESS-CLASS critical )
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 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3129 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3130 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3131 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3134 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3138 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3139 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3141 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3142 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3150 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3151 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3152 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3153 ACCESS-CLASS normal )
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 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3238 ACCESS-CLASS normal )
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 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3263 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3264 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3265 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3266 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3267 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3268 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3269 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3270 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3288 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3294 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3295 ACCESS-CLASS normal )
```

```
ibmattributetypes=( 1.3.18.0.2.4.3296 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3299 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3302 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3303 ACCESS-CLASS restricted )
ibmattributetypes=( 1.3.18.0.2.4.3320 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3321 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3329 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3330 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3331 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3332 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3333 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3334 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3335 ACCESS-CLASS critical )
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 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3396 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3417 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3418 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3419 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3420 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3421 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3422 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3423 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3424 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3425 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3426 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3427 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3428 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3429 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3430 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3431 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3432 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3433 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3434 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3435 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3436 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3437 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3438 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3439 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3440 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3441 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3442 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3443 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3444 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3445 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3446 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3447 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3448 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3449 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3450 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3451 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3452 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3453 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3454 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3455 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3456 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3457 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3458 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3459 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3460 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3461 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3462 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3463 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3464 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3465 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3466 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3467 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3468 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3469 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3470 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3471 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3472 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3473 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3474 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3475 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3476 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3477 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3478 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3479 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3480 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3481 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3482 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3483 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3484 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3485 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3486 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3487 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3488 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3489 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3490 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3491 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3492 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3493 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3494 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3495 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3496 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3497 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3498 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3499 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3500 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3501 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3502 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3503 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3504 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3505 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3506 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3508 ACCESS-CLASS normal )
```

```
ibmattributetypes=( 1.3.18.0.2.4.3509 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3511 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3513 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3514 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3515 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3523 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3524 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3526 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3527 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3528 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3529 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.3559 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.18.0.2.4.3560 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.3656 DBNAME ( 'ibm-slapdServerCompatibilityLevel'
 'ibm-slapdServerCompatibilityLevel' ) ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.454 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.469 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.470 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.473 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.499 ACCESS-CLASS normal )
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 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.10 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.11 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.12 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.13 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.14 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.15 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.16 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.17 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.18 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.19 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.2 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.20 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.21 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.22 ACCESS-CLASS critical )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.3 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.4 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.5 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.6 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.7 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.8 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.42.2.27.8.1.9 ACCESS-CLASS normal )
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 ACCESS-CLASS system )
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 OS/390 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 OS/390 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=( 1.3.18.0.2.6.498  NAME ( 'ibm-slapdTop' )
 DESC 'Global configuration settings for IBM Directory Server on z/OS.' STRUCTURAL SUP ( top $ ibm-slapdConfigEntry )
 MUST ( cn ) MAY ( ibm-slapdServerId $ ibm-slapdMaxPendingChangesDisplayed ) )
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 OS/390 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 OS/390 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 OS/390 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 OS/390 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 OS/390 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 OS/390 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=( 1.3.18.0.2.6.656  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 ) )
objectclasses=( 1.3.18.0.2.6.66  NAME ( 'racfCicsSegment' )
 DESC 'Represents the OS/390 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 OS/390 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 OS/390 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 OS/390 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 OS/390 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 OS/390 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 $
 see Also $ 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. 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 Web site: A Layman's Guide to a Subset of ASN.1, BER, and DER (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, LDAPMDFY, LDAPMDRN, and LDAPDLET utilities have a **-L** option to add this control to LDAP server requests. For more information about the LDAPADD, LDAPMDFY, LDAPMDRN, and LDAPDLET utilities, see Using the LDAP Client Utilities in *z/VM: TCP/IP User's Guide*.

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 will evaluate whether the bound user has permission to modify the candidate attribute values, as determined by the appropriate access controls and the permissions granted by those access controls to the bound DN. If the permissions granted to the bound DN are sufficient to modify the candidate attribute values, those values will be realigned to match their respective new DN values. If any single access check fails, the entire 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 ANSI.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 will cause the control to be ignored. The last time limit value will be 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 will either replace 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 32 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 32 for more information.

  IBMschemaReplaceByValueControl overrides the **schemaReplaceByValue** server configuration option for the current modify request. The last value will be 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 will 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
- **Requested 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 *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 non-zero value to allow paged searches. The **ibm-slapdPagedResAllowNonAdmin** and **ibm-slapdPagedResLmt** dynamic configuration attributes in the **cn=configuration** entry and the **idleConnectionTimeou**t configuration option can be used to limit server resources used by paged searches. The **PersistentSearc**h and **pagedResults** server controls cannot both be specified on a search request. See cn=configuration 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 the 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 already 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: 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. For more information, see *z/VM: TCP/IP User's Guide*.

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 client to create this control and parse the resultant entries. See **ldap_create_persistentsearch_control()** 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 LDBM, CDBM, 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 Configuring the LDAP Server in *z/VM: TCP/IP Planning and Customization* for more information about the **persistentSearch** configuration option.

- **Server behavior:** The server behaves as described in the specification found at IETF documents (datatracker.ietf.org/wg/ipr/documents/), 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 will be skipped if a null-based subtree search is used and the persistent search control is marked as critical, otherwise a typical search will be performed for those backends.

9. If a **PersistentSearch** control is included in a search request for an LDBM, CDBM, 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 time out also affects persistent search connections. See the description of the **idleConnectionTimeout** configuration option in Step 6. Create and Customize the LDAP Configuration File (DS CONF) in *z/VM: TCP/IP Planning and Customization* 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 group 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. 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.

16. 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 will have the same values as on the master.
- **Server behavior:**

1. The control is supported only 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. 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 crtitical, otherwise it ignores the attribute.

4. 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.

5. 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 reside 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 sent only 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 the 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.

  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 the LDAP administrator. If user is not bound as the 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, LDAPMDFY, LDAPMDRN, and LDAPDLET utilities have a **-k** option to add this control to LDAP server requests. For more information about the LDAPADD, LDAPMDFY, LDAPMDRN, and LDAPDLET utilities, see Using the LDAP Client Utilities in *z/VM: TCP/IP User's Guide*.

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 *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* 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 non-zero value to allow sorted searches. The **ibm-slapdSortSrchAllowNonAdmin** and **ibm-slapdSortKeyLimi**t 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 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 3, "LDAP directory schema," on page 13 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 IBM Tivoli Directory Server Client Programming for z/OS and SortKeyRequest 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 .

Appendix C. 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), go to the following web site: A Layman's Guide to a Subset of ASN.1, BER, and DER (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 the 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
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

Error scenario	Account status response
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 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 has to 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 hostname and advanced replication server ID of the supplier server targeted by the extended operation. The format is '*shortenedHostName*:*serverID*'. If the shortened hostname 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

Error scenario	Cascading control replication response
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 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 will be supported.

applicationID ::= 1 for RACF. Other applications will have different identifiers. The identifier informs the LDAP server which (if any) translations of the data should be done.

userid ::= A string containing the userid that 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 userid and group to indicate that the change is to the connection of that user to that group. This string is used to form the value of the **targetDN** attribute in the change log entry.

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.uuuuuuZ*

Where,

yyyy is year, *mm* is month, *dd* is day, *hh* is hour, *ii* is minutes, *ss* is seconds, *uuuuuu* is micro seconds, Z is a character constant meaning that this time is based on Zulu time, also known as GMT.

`initiator` ::= A string containing the userid that made the change. This string is used to form the value of the **ibm-changeInitiatorsName** attribute in the change log entry.

```
changeAttributeList ::= SEQUENCE {
    field         attributeDescription,
    vals           SEQUENCE OF AttributeValue,
    action         ENUMERATED {
                      add (0),
                      replace (1),
                      delete (2) },
    requestValue Boolean }
```

Where,

`field` ::= is the name of the attribute that has been changed. For RACF, this consists of the segment name followed by a period followed by the field name. LDAP maps the RACF segment and field name to an LDAP attribute name.

`vals` ::= is a ber representation (length and data) of the new attribute value.

`action` ::= describes what has happened to the attribute (value add, replace, or delete). To indicate that an entire attribute is deleted, specify an action of `delete` with no value in the `vals` field.

`requestValue` ::= is a flag that, if TRUE, indicates that the attribute value in the `vals` field is not present and should be requested from the application.

The `changeAttributeList` values are used to form the **changes** attribute in the change log entry. If `changeAttributeList` is not specified, a change log entry is created without a **changes** attribute. This acts as a notification to the user of the change log that it should read the entire entry out of the directory tree.

- **Detailed description:** Class and resource cannot be specified with userid 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	changeLogAddEntryRequest's response
An internal error prevents the logging operation from completing	Returns a **loggingFailed** return code

Error scenario	changeLogAddEntryRequest's response
The caller is not in supervisor state	Returns an **invalidCredentials** return code
Change log is not configured	Returns a **notConfigured** return code
Change log is not active	Returns a **notActive** return code
LDAP server is unable to parse the request	Returns a **decodeFailed** return code
Value is outside the range of allowable values	Returns a **valueOutOfRange** return code
LDAP server is unable to convert a RACF userid to an LDAP DN	Returns a **dnConvertFailed** return code

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
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 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

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:** he 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

Error scenario	GetEffectiveACL response
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

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 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

Error scenario	Quiesce or unquiesce context response
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

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 has to 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 hostname and advanced replication server ID of the supplier server targeted by the extended operation. The format is '*shortenedHostName*:*serverID*'. If the shortened hostname 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 server's 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. For more information, see DS2LDIF (ds2ldif utility) in *z/VM: TCP/IP Planning and Customization*.
- **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 file name or dataset 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 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 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 240 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, 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. 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 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 (ds2ldif utility) in *z/VM: TCP/IP Planning and Customization*.

- **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's response
Not bound as the LDAP administrator DN	Returns an **LDAP_OTHER** return code

Error scenario	unloadRequest's response
Multiple LDBM 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 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 6, "Administrative group and roles," on page 91 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	Account status 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 D. Related Protocol Specifications

Many features of TCP/IP for z/VM are based on the following RFCs:

RFC	Title	Author
768	*User Datagram Protocol*	J.B. Postel
791	*Internet Protocol*	J.B. Postel
792	*Internet Control Message Protocol*	J.B. Postel
793	*Transmission Control Protocol*	J.B. Postel
821	*Simple Mail Transfer Protocol*	J.B. Postel
822	*Standard for the Format of ARPA Internet Text Messages*	D. Crocker
823	*DARPA Internet Gateway*	R.M. Hinden, A. Sheltzer
826	*Ethernet Address Resolution Protocol: or Converting Network Protocol Addresses to 48.Bit Ethernet Address for Transmission on Ethernet Hardware*	D.C. Plummer
854	*Telnet Protocol Specification*	J.B. Postel, J.K. Reynolds
856	*Telnet Binary Transmission*	J.B. Postel, J.K. Reynolds
857	*Telnet Echo Option*	J.B. Postel, J.K. Reynolds
877	*Standard for the Transmission of IP Datagrams over Public Data Networks*	J.T. Korb
885	*Telnet End of Record Option*	J.B. Postel
903	*Reverse Address Resolution Protocol*	R. Finlayson, T. Mann, J.C. Mogul, M. Theimer
904	*Exterior Gateway Protocol Formal Specification*	D.L. Mills
919	*Broadcasting Internet Datagrams*	J.C. Mogul
922	*Broadcasting Internet Datagrams in the Presence of Subnets*	J.C. Mogul
950	*Internet Standard Subnetting Procedure*	J.C. Mogul, J.B. Postel
952	*DoD Internet Host Table Specification*	K. Harrenstien, M.K. Stahl, E.J. Feinler
959	*File Transfer Protocol*	J.B. Postel, J.K. Reynolds
974	*Mail Routing and the Domain Name System*	C. Partridge
1009	*Requirements for Internet Gateways*	R.T. Braden, J.B. Postel
1014	*XDR: External Data Representation Standard*	Sun Microsystems Incorporated
1027	*Using ARP to Implement Transparent Subnet Gateways*	S. Carl-Mitchell, J.S. Quarterman
1032	*Domain Administrators Guide*	M.K. Stahl
1033	*Domain Administrators Operations Guide*	M. Lottor
1034	*Domain Names—Concepts and Facilities*	P.V. Mockapetris

RFC	Title	Author
1035	*Domain Names—Implementation and Specification*	P.V. Mockapetris
1042	*Standard for the Transmission of IP Datagrams over IEEE 802 Networks*	J.B. Postel, J.K. Reynolds
1055	*Nonstandard for Transmission of IP Datagrams over Serial Lines: SLIP*	J.L. Romkey
1057	*RPC: Remote Procedure Call Protocol Version 2 Specification*	Sun Microsystems Incorporated
1058	*Routing Information Protocol*	C.L. Hedrick
1091	*Telnet Terminal-Type Option*	J. VanBokkelen
1094	*NFS: Network File System Protocol Specification*	Sun Microsystems Incorporated
1112	*Host Extensions for IP Multicasting*	S. Deering
1118	*Hitchhikers Guide to the Internet*	E. Krol
1122	*Requirements for Internet Hosts-Communication Layers*	R.T. Braden
1123	*Requirements for Internet Hosts-Application and Support*	R.T. Braden
1155	*Structure and Identification of Management Information for TCP/IP-Based Internets*	M.T. Rose, K. McCloghrie
1156	*Management Information Base for Network Management of TCP/IP-based Internets*	K. McCloghrie, M.T. Rose
1157	*Simple Network Management Protocol (SNMP),*	J.D. Case, M. Fedor, M.L. Schoffstall, C. Davin
1179	*Line Printer Daemon Protocol*	The Wollongong Group, L. McLaughlin III
1180	*TCP/IP Tutorial,*	T. J. Socolofsky, C.J. Kale
1183	*New DNS RR Definitions* (Updates RFC 1034, RFC 1035)	C.F. Everhart, L.A. Mamakos, R. Ullmann, P.V. Mockapetris,
1187	*Bulk Table Retrieval with the SNMP*	M.T. Rose, K. McCloghrie, J.R. Davin
1207	*FYI on Questions and Answers: Answers to Commonly Asked Experienced Internet User Questions*	G.S. Malkin, A.N. Marine, J.K. Reynolds
1208	*Glossary of Networking Terms*	O.J. Jacobsen, D.C. Lynch
1213	*Management Information Base for Network Management of TCP/IP-Based Internets: MIB-II,*	K. McCloghrie, M.T. Rose
1215	*Convention for Defining Traps for Use with the SNMP*	M.T. Rose
1228	*SNMP-DPI Simple Network Management Protocol Distributed Program Interface*	G.C. Carpenter, B. Wijnen
1229	*Extensions to the Generic-Interface MIB*	K. McCloghrie
1267	*A Border Gateway Protocol 3 (BGP-3)*	K. Lougheed, Y. Rekhter
1268	*Application of the Border Gateway Protocol in the Internet*	Y. Rekhter, P. Gross

RFC	Title	Author
1269	*Definitions of Managed Objects for the Border Gateway Protocol (Version 3)*	S. Willis, J. Burruss
1293	*Inverse Address Resolution Protocol*	T. Bradley, C. Brown
1270	*SNMP Communications Services*	F. Kastenholz, ed.
1323	*TCP Extensions for High Performance*	V. Jacobson, R. Braden, D. Borman
1325	*FYI on Questions and Answers: Answers to Commonly Asked New Internet User Questions*	G.S. Malkin, A.N. Marine
1351	*SNMP Administrative Model*	J. Davin, J. Galvin, K. McCloghrie
1352	*SNMP Security Protocols*	J. Galvin, K. McCloghrie, J. Davin
1353	*Definitions of Managed Objects for Administration of SNMP Parties*	K. McCloghrie, J. Davin, J. Galvin
1354	*IP Forwarding Table MIB*	F. Baker
1387	*RIP Version 2 Protocol Analysis*	G. Malkin
1389	*RIP Version 2 MIB Extension*	G. Malkin
1393	*Traceroute Using an IP Option*	G. Malkin
1397	*Default Route Advertisement In BGP2 And BGP3 Versions of the Border Gateway Protocol*	D. Haskin
1398	*Definitions of Managed Objects for the Ethernet-like Interface Types*	F. Kastenholz
1440	*SIFT/UFT:Sender-Initiated/Unsolicited File Transfer*	R. Troth
1493	*Definition of Managed Objects for Bridges*	E. Decker, P. Langille, A. Rijsinghani, K. McCloghrie
1540	*IAB Official Protocol Standards*	J.B. Postel
1583	*OSPF Version 2*	J.Moy
1647	*TN3270 Enhancements*	B. Kelly
1700	*Assigned Numbers*	J.K. Reynolds, J.B. Postel
1723	*RIP Version 2 — Carrying Additional Information*	G. Malkin
1738	*Uniform Resource Locators (URL)*	T. Berners-Lee, L. Masinter, M. McCahill
1813	*NFS Version 3 Protocol Specification*	B. Callaghan, B. Pawlowski, P. Stauback, Sun Microsystems Incorporated
1823	*The LDAP Application Program Interface*	T. Howes, M. Smith
2460	*Internet Protocol, Version 6 (IPv6) Specification*	S. Deering, R. Hinden
2052	*A DNS RR for specifying the location of services (DNS SRV)*	A. Gulbrandsen, P. Vixie

RFC	Title	Author
2104	*HMAC: Keyed-Hashing for Message Authentication*	H. Krawczyk, M. Bellare, R. Canetti
2222	*Simple Authentication and Security Layer (SASL)*	J. Myers
2247	*Using Domains in LDAP/X.500 Distinguished Names*	S. Kille, M. Wahl, A. Grimstad, R. Huber, S. Sataluri
2251	*Lightweight Directory Access Protocol (v3)*	M. Wahl, T. Howes, S. Kille
2252	*Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions*	M. Wahl, A. Coulbeck, T. Howes, S. Kille
2253	*Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*	M. Wahl, S. Kille, T. Howes
2254	*The String Representation of LDAP Search Filters*	T. Howes
2255	*The LDAP URL Format*	T. Howes, M. Smith
2256	*A Summary of the X.500 (96) User Schema for use with LDAPv3*	M. Wahl
2279	*UTF-8, a transformation format of ISO 10646*	F. Yergeau
2373	*IP Version 6 Addressing Architecture*	R. Hinden, S. Deering
2461	*Neighbor Discovery for IP Version 6 (IPv6)*	T. Narten, E. Nordmark, W. Simpson
2462	*IPv6 Stateless Address Autoconfiguration*	S. Thomson, T. Narten
2463	*Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*	A. Conta, S. Deering
2710	*Multicast Listener Discovery (MLD) for IPv6*	S. Deering, W. Fenner, B. Haberman
2713	*Schema for Representing Java Objects in an LDAP Directory*	V. Ryan, S. Seligman, R. Lee
2714	*Schema for Representing CORBA Object References in an LDAP Directory*	V. Ryan, R. Lee, S. Seligman
2732	*Format for Literal IPv6 Addresses in URLs*	R. Hinden, B. Carpenter, L. Masinter
2743	*Generic Security Service Application Program Interface Version 2, Update 1*	J. Linn
2744	*Generic Security Service API Version 2 : C-bindings*	J. Wray
2820	*Access Control Requirements for LDAP*	E. Stokes, D. Byrne, B. Blakley, P. Behera
2829	*Authentication Methods for LDAP*	M. Wahl, H. Alvestrand, J. Hodges, R. Morgan
2830	*Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security*	J. Hodges, R. Morgan, M. Wahl
2831	*Using Digest Authentication as a SASL Mechanism*	P. Leach, C. Newman
2849	*The LDAP Data Interchange Format (LDIF)*	G. Good

RFC	Title	Author
2873	*TCP Processing of the IPv4 Precedence Field*	X. Xiao, A. Hannan, V. Paxson, E. Crabble
3377	*Lightweight Directory Access Protocol (v3): Technical Specification*	J. Hodges, R. Morgan
3484	*Default Address Selection for Internet Protocol version 6 (IPv6)*	R. Draves
3513	*Internet Protocol Version 6 (IPv6) Addressing Architecture*	R. Hinden, S. Deering
4191	*Default Router Preferences and More-Specific Routes*	R. Draves, D. Thaler
4517	*LDAP Syntaxes and Matching Rules*	S. Legg
4523	*LDAP Schema Definitions for X.509 Certificates*	K. Zeilenga
5095	*Deprecation of Type 0 Routing Headers in IPv6*	J. Abley, P. Savola, G. Neville-Nei
5175	*IPv6 Router Advertisement Flags Option*	B. Haberman, R. Hinden
5722	*Handling of Overlapping IPv6 Fragments*	S. Krishnan
6946	*Processing of IPv6 "Atomic" Fragments*	F. Gont
6980	*Security Implications of IPv6 Fragmentation with IPv6*	F. Gont

These documents can be obtained from:

Government Systems, Inc.
Attn: Network Information Center
14200 Park Meadow Drive
Suite 200
Chantilly, VA 22021

Many RFCs are available online. Hard copies of all RFCs are available from the NIC, either individually or on a subscription basis. Online copies are available using FTP from the NIC at `nic.ddn.mil`. Use FTP to download the files, using the following format:

```
RFC:RFC-INDEX.TXT
RFC:RFCnnnn.TXT
RFC:RFCnnnn.PS
```

Where:

nnnn
Is the RFC number.

TXT
Is the text format.

PS
Is the PostScript format.

You can also request RFCs through electronic mail, from the automated NIC mail server, by sending a message to `service@nic.ddn.mil` with a subject line of RFC *nnnn* for text versions or a subject line of RFC *nnnn*`.PS` for PostScript versions. To request a copy of the RFC index, send a message with a subject line of RFC INDEX.

For more information, contact `nic@nic.ddn.mil`. Information is also available at Internet Engineering Task Force (www.ietf.org).

Appendix E. Abbreviations and acronyms

The following abbreviations and acronyms are used throughout this book.

Acronym	What it stands for
AIX	Advanced Interactive Executive
ANSI	American National Standards Institute
API	Application Program Interface
APPC	Advanced Program-to-Program Communications
APPN	Advanced Peer-to-Peer Networking
ARP	Address Resolution Protocol
ASCII	American National Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
AUI	Attachment Unit Interface
BFS	Byte File System
BIOS	Basic Input/Output System
BNC	Bayonet Neill-Concelman
CCITT	Comite Consultatif International Telegraphique et Telephonique (The International Telegraph and Telephone Consultative Committee)
CLIST	Command List
CMS	Conversational Monitor System
CP	Control Program
CPI	Common Programming Interface
CREN	Corporation for Research and Education Networking
CSD	Corrective Service Diskette
CTC	Channel-to-Channel
CU	Control Unit
CUA	Common User Access
DASD	Direct Access Storage Device
DBCS	Double Byte Character Set
DLL	Dynamic Link Library
DNS	Domain Name System
DOS	Disk Operating System
DPI	Distributed Program Interface
EBCDIC	Extended Binary-Coded Decimal Interchange Code
EISA	Enhanced Industry Standard Adapter
ESCON	Enterprise Systems Connection Architecture
FAT	File Allocation Table

Abbreviations and Acronyms

Acronym	What it stands for
FTAM	File Transfer Access Management
FTP	File Transfer Protocol
FTP API	File Transfer Protocol Applications Programming Interface
GCS	Group Control System
GDF	Graphics Data File
HPFS	High Performance File System
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IP	Internet Protocol
IPL	Initial Program Load
ISA	Industry Standard Adapter
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
IUCV	Inter-User Communication Vehicle
JES	Job Entry Subsystem
JIS	Japanese Institute of Standards
JCL	Job Control Language
LAN	Local Area Network
LAPS	LAN Adapter Protocol Support
LDAP	Lightweight Directory Access Protocol
LPQ	Line Printer Query
LPR	Line Printer Client
LPRM	Line Printer Remove
LPRMON	Line Printer Monitor
LU	Logical Unit
MAC	Media Access Control
Mbps	Megabits per second
MBps	Megabytes per second
MCA	Micro Channel Adapter
MIB	Management Information Base
MIH	Missing Interrupt Handler
MILNET	Military Network
MHS	Message Handling System
MTU	Maximum Transmission Unit
MVS	Multiple Virtual Storage

Acronym	What it stands for
MX	Mail Exchange
NCP	Network Control Program
NDIS	Network Driver Interface Specification
NFS	Network File System
NIC	Network Information Center
NLS	National Language Support
NSFNET	National Science Foundation Network
OS/2	Operating System/2®
OSA	Open Systems Adapter
OSF	Open Software Foundation, Inc.
OSI	Open Systems Interconnection
OSIMF/6000	Open Systems Interconnection Messaging and Filing/6000
OV/MVS	OfficeVision/MVS
OV/VM	OfficeVision/VM
PAD	Packet Assembly/Disassembly
PC	Personal Computer
PCA	Parallel Channel Adapter
PDN	Public Data Network
PDU	Protocol Data Units
PING	Packet Internet Groper
PIOAM	Parallel I/O Access Method
POP	Post Office Protocol
PROFS	Professional Office Systems
PSCA	Personal System Channel Attach
PSDN	Packet Switching Data Network
PU	Physical Unit
PVM	Passthrough Virtual Machine
RACF	Resource Access Control Facility
RARP	Reverse Address Resolution Protocol
REXEC	Remote Execution
REXX	Restructured Extended Executor Language
RFC	Request For Comments
RIP	Routing Information Protocol
RISC	Reduced Instruction Set Computer
RPC	Remote Procedure Call
RSCS	Remote Spooling Communications Subsystem
SAA	Systems Application Architecture®

Abbreviations and Acronyms

Acronym	What it stands for
SBCS	Single Byte Character Set
SFS	Shared File System
SLIP	Serial Line Internet Protocol
SMIL	Structure for Management Information
SMTP	Simple Mail Transfer Protocol
SNA	Systems Network Architecture
SNMP	Simple Network Management Protocol
SOA	Start of Authority
SPOOL	Simultaneous Peripheral Operations Online
SQL	IBM Structured Query Language
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TSO	Time Sharing Option
TTL	Time-to-Live
UDP	User Datagram Protocol
VGA	Video Graphic Array
VM	Virtual Machine
VMCF	Virtual Machine Communication Facility
VM/ESA	Virtual Machine/Enterprise System Architecture
VMSES/E	Virtual Machine Serviceability Enhancements Staged/Extended
VTAM®	Virtual Telecommunications Access Method
WAN	Wide Area Network
XDR	eXternal Data Representation

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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 grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

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

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 jurisdictions 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide 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:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

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 document 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.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information may 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.

Programming Interface Information

This book primarily documents information that is NOT intended to be used as Programming Interfaces of z/VM.

This book also documents intended Programming Interfaces that allow the customer to write programs to obtain services of z/VM. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

`PI`

<....Programming Interface information....>

`PI end`

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., in the United States and/or other countries. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on IBM Copyright and trademark information (https://www.ibm.com/legal/copytrade).

Adobe and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java is a trademark or registered trademark of Oracle and/or its affiliates.

The registered trademark Linux is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM online privacy statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see:

- The section entitled **IBM Websites** at IBM Privacy Statement (https://www.ibm.com/privacy)
- Cookies and Similar Technologies (https://www.ibm.com/privacy#Cookies_and_Similar_Technologies)

Bibliography

This topic lists the publications in the z/VM library. For abstracts of the z/VM publications, see *Abstracts of the z/VM base product publications* and *Abstracts of the z/VM optional feature publications* in *z/VM: General Information*.

Where to get z/VM information

The current z/VM product documentation is available in IBM Documentation - z/VM (https://www.ibm.com/docs/en/zvm).

For lists of additional sources of information about z/VM, see *Where to get information about z/VM* in *z/VM: General Information*.

z/VM base library

Overview

- *z/VM: License Information*, GI13-4377
- *z/VM: General Information*, GC24-6286

Installation, migration, and service

- *z/VM: Installation Guide*, GC24-6292
- *z/VM: Migration Guide*, GC24-6294
- *z/VM: Service Guide*, GC24-6325
- *z/VM: VMSES/E Introduction and Reference*, GC24-6336

Planning and administration

- *z/VM: CMS File Pool Planning, Administration, and Operation*, SC24-6261
- *z/VM: CMS Planning and Administration*, SC24-6264
- *z/VM: Connectivity*, SC24-6267
- *z/VM: CP Planning and Administration*, SC24-6271
- *z/VM: Getting Started with Linux on IBM Z*, SC24-6287
- *z/VM: Group Control System*, SC24-6289
- *z/VM: I/O Configuration*, SC24-6291
- *z/VM: Running Guest Operating Systems*, SC24-6321
- *z/VM: Saved Segments Planning and Administration*, SC24-6322
- *z/VM: Secure Configuration Guide*, SC24-6323

Customization and tuning

- *z/VM: CP Exit Customization*, SC24-6269
- *z/VM: Performance*, SC24-6301

Operation and use

- *z/VM: CMS Commands and Utilities Reference*, SC24-6260
- *z/VM: CMS Primer*, SC24-6265

- *z/VM: CMS User's Guide*, SC24-6266
- *z/VM: CP Commands and Utilities Reference*, SC24-6268
- *z/VM: System Operation*, SC24-6326
- *z/VM: Virtual Machine Operation*, SC24-6334
- *z/VM: XEDIT Commands and Macros Reference*, SC24-6337
- *z/VM: XEDIT User's Guide*, SC24-6338

Application programming

- *z/VM: CMS Application Development Guide*, SC24-6256
- *z/VM: CMS Application Development Guide for Assembler*, SC24-6257
- *z/VM: CMS Application Multitasking*, SC24-6258
- *z/VM: CMS Callable Services Reference*, SC24-6259
- *z/VM: CMS Macros and Functions Reference*, SC24-6262
- *z/VM: CMS Pipelines User's Guide and Reference*, SC24-6252
- *z/VM: CP Programming Services*, SC24-6272
- *z/VM: CPI Communications User's Guide*, SC24-6273
- *z/VM: ESA/XC Principles of Operation*, SC24-6285
- *z/VM: Language Environment User's Guide*, SC24-6293
- *z/VM: OpenExtensions Advanced Application Programming Tools*, SC24-6295
- *z/VM: OpenExtensions Callable Services Reference*, SC24-6296
- *z/VM: OpenExtensions Commands Reference*, SC24-6297
- *z/VM: OpenExtensions POSIX Conformance Document*, GC24-6298
- *z/VM: OpenExtensions User's Guide*, SC24-6299
- *z/VM: Program Management Binder for CMS*, SC24-6304
- *z/VM: Reusable Server Kernel Programmer's Guide and Reference*, SC24-6313
- *z/VM: REXX/VM Reference*, SC24-6314
- *z/VM: REXX/VM User's Guide*, SC24-6315
- *z/VM: Systems Management Application Programming*, SC24-6327
- *z/VM: z/Architecture Extended Configuration (z/XC) Principles of Operation*, SC27-4940

Diagnosis

- *z/VM: CMS and REXX/VM Messages and Codes*, GC24-6255
- *z/VM: CP Messages and Codes*, GC24-6270
- *z/VM: Diagnosis Guide*, GC24-6280
- *z/VM: Dump Viewing Facility*, GC24-6284
- *z/VM: Other Components Messages and Codes*, GC24-6300
- *z/VM: VM Dump Tool*, GC24-6335

z/VM facilities and features

Data Facility Storage Management Subsystem for z/VM

- *z/VM: DFSMS/VM Customization*, SC24-6274
- *z/VM: DFSMS/VM Diagnosis Guide*, GC24-6275

- *z/VM: DFSMS/VM Messages and Codes*, GC24-6276
- *z/VM: DFSMS/VM Planning Guide*, SC24-6277
- *z/VM: DFSMS/VM Removable Media Services*, SC24-6278
- *z/VM: DFSMS/VM Storage Administration*, SC24-6279

Directory Maintenance Facility for z/VM

- *z/VM: Directory Maintenance Facility Commands Reference*, SC24-6281
- *z/VM: Directory Maintenance Facility Messages*, GC24-6282
- *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*, SC24-6283

Open Systems Adapter

- Open Systems Adapter-Express Customer's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/ioa2z1f0.pdf), SA22-7935
- Open Systems Adapter Integrated Console Controller User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC27-9003-02.pdf), SC27-9003
- Open Systems Adapter-Express ICC 3215 Support (https://www.ibm.com/docs/en/zos/2.3.0?topic=osa-icc-3215-support), SA23-2247
- Open Systems Adapter/Support Facility on the Hardware Management Console (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC14-7580-02.pdf), SC14-7580

Performance Toolkit for z/VM

- *z/VM: Performance Toolkit Guide*, SC24-6302
- *z/VM: Performance Toolkit Reference*, SC24-6303

RACF Security Server for z/VM

- *z/VM: RACF Security Server Auditor's Guide*, SC24-6305
- *z/VM: RACF Security Server Command Language Reference*, SC24-6306
- *z/VM: RACF Security Server Diagnosis Guide*, GC24-6307
- *z/VM: RACF Security Server General User's Guide*, SC24-6308
- *z/VM: RACF Security Server Macros and Interfaces*, SC24-6309
- *z/VM: RACF Security Server Messages and Codes*, GC24-6310
- *z/VM: RACF Security Server Security Administrator's Guide*, SC24-6311
- *z/VM: RACF Security Server System Programmer's Guide*, SC24-6312
- *z/VM: Security Server RACROUTE Macro Reference*, SC24-6324

Remote Spooling Communications Subsystem Networking for z/VM

- *z/VM: RSCS Networking Diagnosis*, GC24-6316
- *z/VM: RSCS Networking Exit Customization*, SC24-6317
- *z/VM: RSCS Networking Messages and Codes*, GC24-6318
- *z/VM: RSCS Networking Operation and Use*, SC24-6319
- *z/VM: RSCS Networking Planning and Configuration*, SC24-6320

TCP/IP for z/VM

- *z/VM: TCP/IP Diagnosis Guide*, GC24-6328

- *z/VM: TCP/IP LDAP Administration Guide*, SC24-6329
- *z/VM: TCP/IP Messages and Codes*, GC24-6330
- *z/VM: TCP/IP Planning and Customization*, SC24-6331
- *z/VM: TCP/IP Programmer's Reference*, SC24-6332
- *z/VM: TCP/IP User's Guide*, SC24-6333

Prerequisite products

Device Support Facilities

- Device Support Facilities (ICKDSF): User's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_3.1.0/pdf/ickug00_v3r1.pdf), GC35-0033

Related Products

XL C++ for z/VM

- *XL C/C++ for z/VM: Runtime Library Reference*, SC09-7624
- *XL C/C++ for z/VM: User's Guide*, SC09-7625

z/OS

IBM Documentation - z/OS (https://www.ibm.com/docs/en/zos)

Other TCP/IP Related Publications

This section lists other publications, outside the z/VM 7.4 library, that you may find helpful.

- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *TCP/IP Illustrated, Volume 1: The Protocols*, SR28-5586
- *Internetworking with TCP/IP Volume I: Principles, Protocols, and Architecture*, SC31-6144
- *Internetworking With TCP/IP Volume II: Implementation and Internals*, SC31-6145
- *Internetworking With TCP/IP Volume III: Client-Server Programming and Applications*, SC31-6146
- *DNS and BIND in a Nutshell*, SR28-4970
- "MIB II Extends SNMP Interoperability," C. Vanderberg, *Data Communications*, October 1990.
- *"Network Management and the Design of SNMP,"* J.D. Case, J.R. Davin, M.S. Fedor, M.L. Schoffstall.
- *"Network Management of TCP/IP Networks: Present and Future,"* A. Ben-Artzi, A. Chandna, V. Warrier.
- "Special Issue: Network Management and Network Security,"*ConneXions-The Interoperability Report*, Volume 4, No. 8, August 1990.
- *The Art of Distributed Application: Programming Techniques for Remote Procedure Calls*, John R. Corbin, Springer-Verlog, 1991.
- *The Simple Book: An Introduction to Management of TCP/IP-based Internets*, Marshall T Rose, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

Index

Special Characters

_passwd ()
    errno values returned by 62
' (apostrophe) 10
" (double quote) support in SDBM 70
\ (backslash) support in SDBM 70
(pound sign) 10
(pound sign) support in SDBM 70
+ (plus sign) 10
+ (plus sign) support in SDBM 70
= (equal sign) 10
> (greater than sign) 10

A

abandon behavior 299
abbreviations and acronyms 383
ABSTRACT object class type 20, 31
access
    determining 155
access classes
    attribute 148
    determining 146
    permissions 149
access control
    attributes 145
    groups 165
    LDAP server capability 5
    using 145
    using RACF 61
access control and ownership
    modify DN 44
Access Control List (ACL)
    aclEntry attribute 146
    aclPropagate attribute 150
    aclSource attribute 150
    attribute classes 161
    creating a group for 174
    creating owner for entry 171
    deleting owner for entry 174
    description 4, 145
    entryOwner attribute 150
    examples 165
    filters 161
    groups 165
    information, retrieving 167
    modifying owner for entry 172
    override example 164
    ownerPropagate attribute 151
    ownerSource attribute 151
    propagation 151, 164
    requested attributes 161
    searching 161
Account status extended operation example 127
acctstatus 122
ACL (Access Control List)

ACL (Access Control List) *(continued)*
    See Access Control List (ACL) 4
ACL attributes
    entry owner attributes 145
ACL filters 151
ACL restrictions group gathering 136
acl restrictions group membership 136
aclEntry attribute
    description 146
aclEntry syntax 146
aclPropagate attribute
    description 150
aclSource attribute
    description 150
Activating password policy 104
adding, modifying, deleting group entries examples 137
administration
    restricting access 4
administration group and roles 91
administrative group and roles-related extended operation 100
administrative group member examples 98
administrative roles 91
administrative roles and extended operations 99
administrative roles defined in LDAP 95
administrative roles defined in RACF 97
advanced replication 5
Advanced replication
    replication 193
Advanced replication configuration examples 217
Advanced replication error recovery example 252
Advanced replication features 199
Advanced replication maintenance mode 239
Advanced replication overview 196
Advanced replication related entries summary 217
Advanced replication related extended operations 244
Advanced replication terminology 193
alias
    description 257
alias entry 257
aliases
    LDAP server capability 5
aliasing on search performance
    search performance 257
analyzing
    schema errors 35
anonymous searches 161
apostrophe 10
arranging information 2
Associating
    LDAP attributes 63
attribute
    object class 3
    syntaxes 22
attribute classes
    searching 161
Attribute encryption

Attribute encryption *(continued)*
 description 7
attribute types
 description 15
 format 29
 usage 29
attributes
 access allowed for 146
 access classes 146, 148
 access control 145
 aclEntry 146
 aclPropagate 150
 aclSource 150
 cn 180
 deleting in SDBM 88
 description 180
 determining 9
 entryOwner 150
 ibm-entryCheckSum 8
 ibm-entryCheckSumOp 8
 ibm-entryuuid 8
 jpegPhoto 2
 LDAP schema 23
 mail 2
 mandatory for replica entry 179
 multi-valued ref 273
 multi-valued with RACF 77
 normalizing 299
 optional for replica entry 180
 ownerpropagate 151
 ownerSource 151
 ref 273
 replicaBindDN 179
 replicaBindMethod 181
 replicaCredentials 179
 replicaHost 179
 replicaPort 180
 replicaUpdateTimeInterval 180
 replicaUseSSL 180
 requested 161
 returning lowercase 299
 searching 161
 seeAlso 180
 syntaxes 9
 type 14
attributeTypes schema attribute 20, 30
authenticateOnly server control 345
authentication
 client 6
 server 6
authentication bind
 CRAM-MD5 and DIGEST-MD5 6
AUXILIARY object class type 20, 31

B

backend
 multiple 4
 referral entries 274
backslash (\) support in SDBM 70
backslash character
 DN syntax 10
backup of master server 177
basic replication

basic replication *(continued)*
 associating servers with 277
 LDAP server capability 5
 SSL 187
basic replication replicas
 periodic validation, compatible server versions 58
benefits of replication 177
bind, SASL 6
binding
 authenticateOnly server control 345
Binding with SDBM using password policy 62
blank spaces
 using in DNs 10
building
 directory namespace 286

C

cache tuning 304
capabilities of LDAP server 4
CDBM 5
CDBM backend
 attribute types supported 29
 IBM attribute types supported 30
 initializing ACLs 154
 matching rules supported 25
 object class supported 30
 RDN 9
CDBM schema
 setting up 13
certificate
 client 6
certificate management 301
change log
 additional required configuration 266
 set up and using LDAP server for logging changes 270
 when changed are logged 267
change log entries 268
change log information root DSE entry 270
change log schema 267
change logging
 LDAP server capability 5
Changing password values when pwdsafemodify is set to true 127
classes, access
 attribute 148
 determining 146
 permissions 149
 specifying for LDBM 20
client, LDAP
 See LDAP client 293
complex modify DN replication 178
concurrent database instances 4
configuration backen 5
configuration file
 default referral 275
 master server 188
 setting SSL keywords 188
configuring
 LDAP server capability 5
 replica server 184
configuring file-based GDBM backend
 file-based GDBM backend 266
configuring GDBM backend

IBM attribute types *(continued)*
    description 20
    format 30
    usage 30
ibm-entryuuid
    replication 177
ibm-entryuuid,attribute 8
ibm-slapdAdminGroupMember objectclass attributes
    entry attribute descriptions
        ibm-slapdAdminDN 96
        ibm-slapdAdminPW 96
        ibm-slapdAdminRole 97
IBMAttributeTypes schema attribute 20, 30
IBMSchemaReplaceByValue 346
IBMSchemaReplaceByValueControl 346
Individual password policy example 125
information
    arranging 2
    layout 285
    protecting 4
    referencing 3
inheritance
    default 151
initial schema
    for LDAP server 323
Initial validation of compatible server versions
    consumer and replica servers 58
initializing
    replica directory 183
Initializing ACLs with schema entry
    Initializing ACLs 154
international characters
    retrieving 6
    storing 6

J

JPEG format 2

K

key management 301

L

lA5 character set 299
large static groups 316
layout, information 285
LDAP (Lightweight Directory Access Protocol)
    See Lightweight Directory Access Protocol (LDAP) 2
LDAP 2
    referrals 276
    referrals, limitations 276
LDAP 3
    referrals 276
    UTF-8 characters 299
LDAP 3 support 6
LDAP client
    authenticateOnly server control 345
    considerations 293
    using in LDAP 4
    UTF-8 data 299
LDAP daemon

LDAP daemon *(continued)*
    See LDAP server 2
LDAP Data Interchange Format (LDIF)
    description 288
LDAP directory
    protecting information in 4
LDAP schema attributes 23
ldap server
    threads 303
LDAP server
    access control 145
    attribute types supported 29
    authenticateOnly server control 345
    capabilities 4
    changing replica to master 185
    data model 9
    example configuration 286
    extended group membership searching 6
    IBM attribute types supported 30
    LDAP 3 support 6
    LDAP syntaxes supported 23
    master and replica 187
    matching rules supported 25
    model for 2
    naming 4
    object classes supported 30
    overview 1
    RACF 61
    replica 183
    replication 177
    SDBM backend 61
    using 4
LDAP syntaxes
    description 21
    format 23
    supported, general use 23
    supported, server use 25
    usage 23
LDBM backend
    access control
        pseudo DN 147
    adding replica entries 181
    attribute types supported 29
    CRAM-MD5 and DIGEST-MD5 129
    Default ACLs 153
    IBM attribute types supported 30
    initializing ACLs 153
    matching rules supported 25
    object class supported 30
    RDN 9
LDBM schema
    setting up 13
LDIF files
    editing 288
LE heap pool considerations 318
less than sign 10
Lightweight Directory Access Protocol (LDAP)
    description 2
    how it works 4
    schema publication 13
limitations, referrals 276
loading
    directory information 288
locating

locating *(continued)*
  subschemaSubentry DN 36

M

mail attribute 2
maintenance mode
  peer to peer 182
manageDsaIT 277
manageDsaIT server control 345–347
managing
  replication 243
managing group search limits 136
master
  backup of 177
  changing replica to 185
  communicating with replica 187
  database, description 177
  server 177
  server, setting up 188
  using replication 291
Master-replica replication 196
matching rules
  description 22
  EQUALITY values 16
  EQUALITY, defaults 15
  format 25
  ORDERING values 16
  SUBSTR values 16
  supported 26
  usage 25
MAY attribute type 20, 31
mdoify DN
  access control 44
membership
  extended group, searching 6
minimum schema
  for LDBM 323
modify
  updating ACLs 167
modify DN
  access control changes 46
  basic replication synchronization 59
  complex replication 178
  considerations 44
  considerations in the use of 42
  entries for rename 43
  operation syntax 37
  operations and replication 57
  relocating an entry 45, 46
  scenario constraints 50
  SDBM schema 69
  suffix DNs 49
Modify DN
  validation 58
monitor search examples 313
monitor support 299
monitoring performance 307
multi-valued ref attribute 273
multiple databases
  concurrent 4
multiple directories
  replication of 177
multiple socket ports 7

MUST attribute type 20, 31

N

namespace
  directory 285
  entries, RACF 68
  hierarchy diagram for RACF 69
native authentication
  capability 6
nested groups 135
no administrator 92
normal access class 148
numeric object identifier (OID) 31

O

object class
  adding for CDBM 30
  adding for LDBM 30
  CDBM usage 30
  definitions, adding for CDBM 30
  definitions, adding for LDBM 30
  description 9, 20
  extensibleObject 31
  format for CDBM 30
  format for LDBM 30
  hierarchy 21
  LDBM usage 30
  person 285
  referral 273
  replicaObject 179
object class attribute
  description 3
object class definitions
  adding for CDBM 30
  adding for LDBM 30
  specifying for CDBM 30
  specifying for LDBM 30
object identifiers
  oid
    supported and enabled capabilities 295
objectClass attribute 14
objects
  protecting 145
oedit editor 288
OID (numeric object identifier) 31
operational administrator 92
operational attributes in user entries 116–118
operations
  defining 4
organizing
  directory namespace 285
  information 2
out-of-sync conditions 190
overhead, reducing 4
override, ACL example 164
Overriding password policy and unlocking accounts 122
owner
  creating for entry 171
  deleting for entry 174
  modifying for entry 172
  ownerPropagate attribute 151

querying *(continued)*
    subschemaSubentry 14
querying group membership examples 140

R

racfAttributes
    racfConnectAttributes 68
RDN (relative distinguished name)
    See relative distinguished name (RDN) 3
read-only replica
    Upgrading to be a peer replica of the master server 187
recovering
    from out-of-sync conditions 190
Recovering from advanced replication errors 249
ref attribute 273
references
    setup, recommended 274
referencing information 3
referrals
    default 287
    default, defining 275
    description 273, 290
    example of distributing namespace 277
    LDAP 2 276
    LDAP 3 276
    LDAP server capability 5
    limitations with LDAP 2 276
    manageDsaIT server control 345–347
    object 290
    processing 275
    replication 185
    specifying 287
    suppressing 345–347
related protocols 377
relative distinguished name (RDN)
    description 3, 9
relocating an entry with DN realignment 46
Replacing individual schema values
    Replacing values 33
replica
    changing to master 185
    communicating with master 187
    entry 188
    setting up 188, 291
Replica groups 201
Replica server with SSL/TLS enablement 242
Replica subentries 202
replicateOperational 352
replicateOperationalAttributes 352
replicating
    server 179
replicating password policy operational attributes 120
Replicating server with SSL/TLS enablement 242
replication
    benefits 177
    command line tasks
        configuration information 245
    database,description 177
    description 177
    entries 179
    entries, adding in LDBM 181
    ibm-entryuuid 177
    partial replication 240

replication *(continued)*
    password encryption 178
    server 183
    server, configuring 184
    setting up for 290
    troubleshooting 189
replication administrator 93
Replication agreements 203
replication bind failure time stamp control 353
Replication conflict resolution 199
Replication contexts 201
Replication scheduling 199
Replication schema updates 238
Replication topology 195
Replication topology hints and tips 237
requested attributes 161
Resource Access Control Facility (RACF)
    changing password or password phrase with SDBM 83
    configuring LDAP server for 4
    connection to group 76
    distinguished names 10
    LDAP access to 61
    namespace entries 68
    RACF fields 63
    univeral groups
        searching 82
retrieving racf
    racf password envelope
        user password or password phrase envelopes 83
robust general-purpose databases 4
root administrator 94
root DSE
    searching 293
    support of 6
rules, matching
    See matching rules 25
running
    LDAP tools with SDBM 70

S

samples
    object class hierarchy 21
    schema entry 14
SASL CRAM-MD5 and DIGEST-MD5 6
SASL external bind 6
Schedule entries 208
schema
    dynamic 6
    initial for LDAP server 323
    minimum for LDBM 323
schema (CDBM)
    attribute syntax 22
    attribute types 29
    defining new elements 31
    errors, analyzing 35
    IBM attribute types 30
    LDAP attributes 23
    LDAP syntaxes supported 23
    matchng rules 25
    object classes 30
    sample entry 14
    updating CDBM 32
schema (LDBM)

T

things to consider 215
threads
    for performance 4
time stamp control 353
trademarks 388
tree structure
    hierarchical 2
trimming change log 269
troubleshooting
    replication 189
types of information to store 2

U

Unconfiguring advanced replication 238
universal groups
    searching 82
unloading and loading change log 269
Unlocking or unexpiring the LDAP administrator's account
123
unsynchronized directories 190
update, CDBM schema 32
update, LDBM schema 32
updating
    schema errors 35
Updating a numeric object identifier (NOID)
    Updating a NOID
        NOID 35
user Attribute encryption
    See Attribute encryption 7
user groups considerations in large directories 315
User type 375
user type extended operation examples 100
using the limits from search limit groups 137
UTC Time syntax 35
UTF-8 characters
    retrieving 6
    storing 6, 299
utilities
    LDAP operation
        using with SDBM 84

V

ValueControl
    ibmSchema 346

W

white space 10

X

X.500
    data model 9
    description 4

IBM

Product Number: 5741-A09

Printed in USA