z/VM

IBM

TCP/IP LDAP Administration Guide

version 6 release 1

SC24-6236-00

z/VM

TCP/IP LDAP Administration Guide

version 6 release 1

SC24-6236-00

Note:
Before using this information and the product it supports, read the information under "Notices" on page 235.

This edition applies to version 6, release 1, modification 0 of IBM z/VM (product number 5741-A07) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC24-6140-01.

Contents

About this document

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

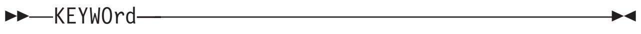
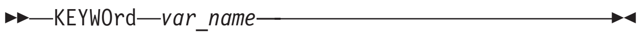
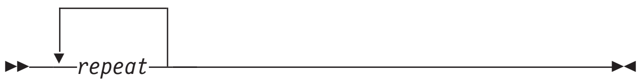
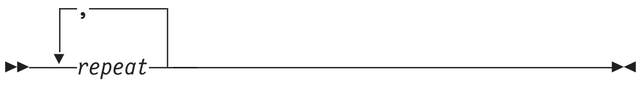
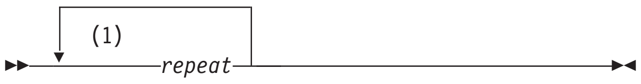
This section describes how to read the syntax diagrams in this document.

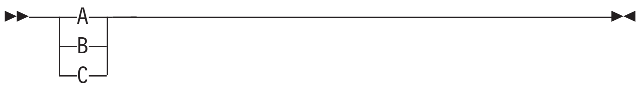
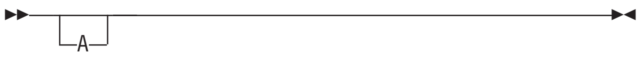
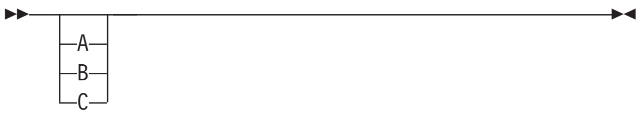
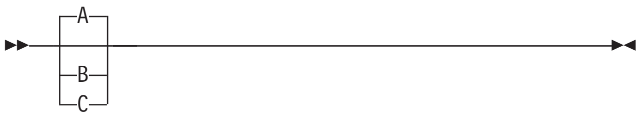
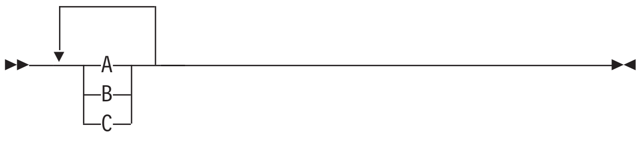
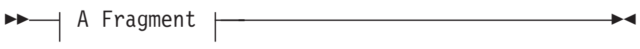
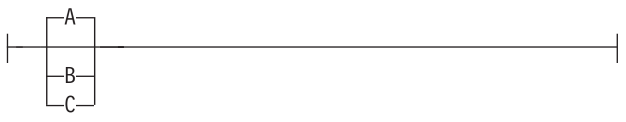
Getting Started: To read a syntax diagram, follow the path of the line. Read from left to right and top to bottom.

- 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: 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.	
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.	
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.	
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.	
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.	
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:** 

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 D, "Abbreviations and Acronyms," on page 231, lists the abbreviations and acronyms that are used throughout this document.

The “Glossary” on page 239, defines terms used throughout this document that are associated with TCP/IP communication in an internet environment.

For more information about related publications, see the documents listed in the “Bibliography” on page 241.

Links to Other Online Documents

If you are viewing the Adobe® Portable Document Format (PDF) version of this document, it might contain links to other documents. A link to another document is based on the name of the requested PDF file. The name of the PDF file for an IBM document is unique and identifies the edition. The links provided in this document are for the editions (PDF names) that were current when the PDF file for this document was generated. However, newer editions of some documents (with different PDF names) might exist. A link from this document to another document works only when both documents reside in the same directory.

How to send your comments to IBM

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

Use one of the following methods to send us your comments:

1. Send an e-mail to mhvrcfs@us.ibm.com
2. Visit the z/VM reader's comments Web page at www.ibm.com/systems/z/os/zvm/zvmforms/webqs.html
3. Mail the comments to the following address:
   IBM Corporation
   Attention: MHVRCFS Reader Comments
   Department H6MA, Mail Station P181
   2455 South Road
   Poughkeepsie, NY 12601-5400
   U.S.A.
4. Fax the comments to us as follows:
   From the United States and Canada: 1+845+432-9405
   From all other countries: Your international access code +1+845+432-9405

Include the following information:

- Your name and address
- Your e-mail address
- Your telephone or fax number
- The publication title and order number:
  z/VM V6R1 TCP/IP LDAP Administration Guide
  SC24-6236-00
- The topic and page number related to your comment
- The text of your comment

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

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

If you have a technical problem

Do not use the feedback methods listed above. Instead, do one of the following:

- Contact your IBM service representative.
- Contact IBM technical support.
- Visit the z/VM support Web page at www.vm.ibm.com/service/
- Visit the IBM mainframes support Web page at www.ibm.com/systems/support/z/

Summary of changes

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the changes. Some program updates might be provided through z/VM service by program temporary fixes (PTFs) for authorized program analysis reports (APARs), which also might be available for some prior releases.

SC24-6236-00, z/VM Version 6 Release 1

This edition includes changes or additions to support the general availability of z/VM V6.1. For this edition, the following changes have been made:

- A clarification was added to the LDAP Program Call support. See “Additional required configuration” on page 146.
- Reference information for the gskkyman and gsktrace utilities has been moved to *z/VM: TCP/IP User’s Guide*.

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:
- Interoperability with any Version 2 or Version 3 LDAP directory client
- Access controls on directory information, using static, dynamic, and nested groups
- Secure Sockets Layer (SSL) communication (SSL V3 and TLS V1)
- Start TLS (Transport Layer Security) activation of secure communication
- Client and server authentication using SSL/TLS
- Password encryption
- Replication
- Referrals
- Aliases
- Change logging
- LDAP Version 2 and Version 3 protocol support
- Schema publication and update
- Native authentication
- CRAM-MD5 (Challenge-Response Authentication Method) and DIGEST-MD5 authentication
- Root DSE information
- LDAP access to information stored in RACF®
- 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 LDAP, which stands for the Lightweight Directory Access Protocol. LDAP Version 2 (V2) and LDAP Version 3 (V3), both supported in z/VM, are directory service protocols that run over TCP/IP. The details of LDAP V2 are defined in Internet Engineering Task Force (IETF) Request for Comments (RFC) 1777, *The Lightweight Directory Access Protocol*, and the details of LDAP V3 are defined in IETF RFCs 2251 through 2256. For a list of supported RFCs, see Appendix C, "Related Protocol Specifications," on page 225.

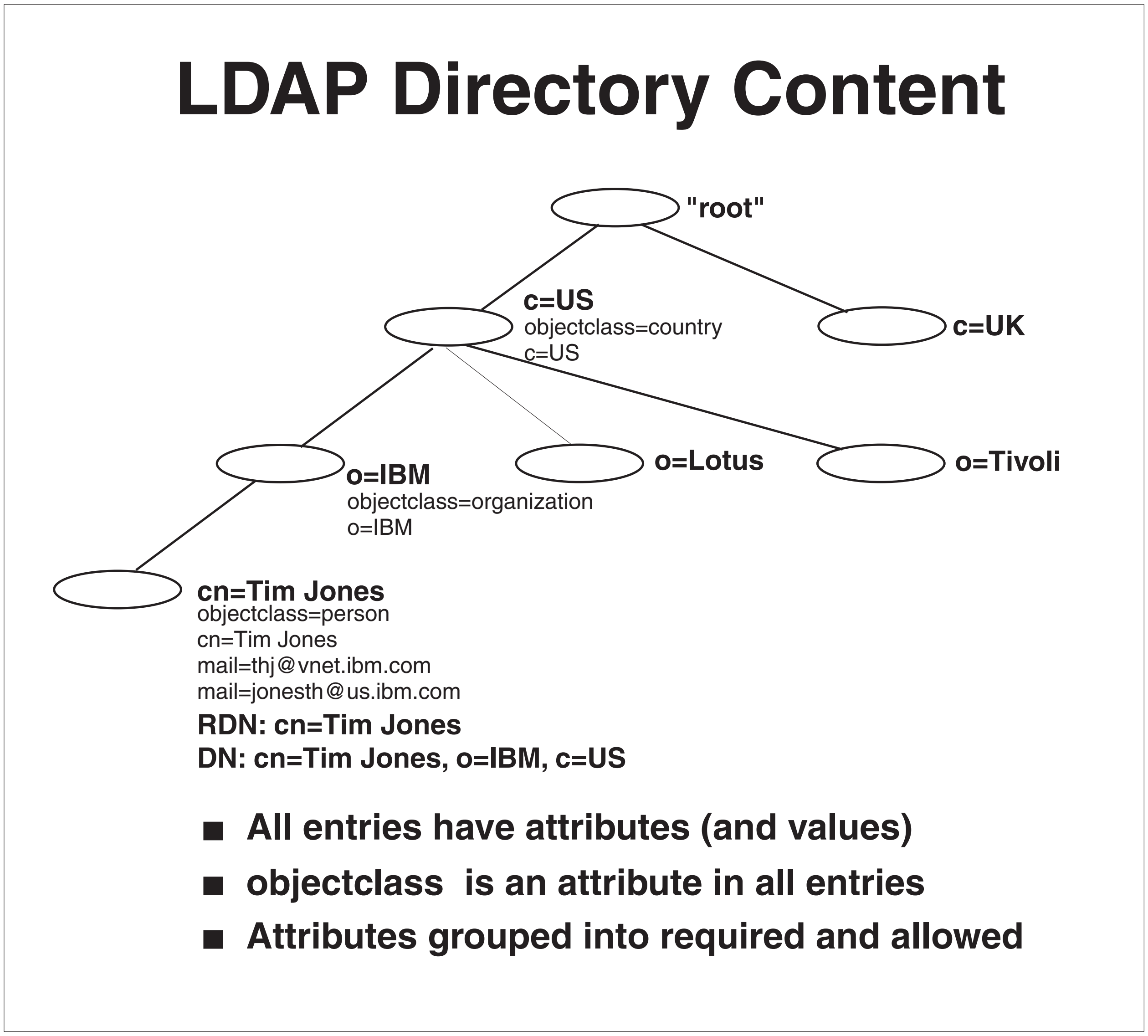
This section gives an overview of LDAP from a user's perspective.

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, which should help make things clear.



LDAP Directory Content

- **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=IBM, 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=ibm,dc=com`. These naming formats can be mixed as well, for example: `cn=Tim Brown,ou=Sales,dc=ibm,dc=com`.

How is the information accessed?

LDAP defines operations for interrogating and updating the directory. Operations are provided for adding 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?" 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 OpenExtensions™ files.
- **Access to RACF data**. The LDAP server can be configured to provide read/write access to RACF user, group, and connection profiles using the LDAP protocol. 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 59.

  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, and connection profile information, RACF is required.
- **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 8, "Using access control," on page 97. For more information about groups, see Chapter 7, "Static, dynamic, and nested groups," on page 85.
- **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.
- **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 9, "Replication," on page 121. This feature is contrasted with multiple concurrent servers.
- **Referrals**. The LDAP server provides the ability to refer clients to additional directory servers. Using referrals you can distribute processing overhead, distribute administration of data along organizational boundaries, and provide potential for widespread interconnection beyond an organization's own boundaries. For more information, see Chapter 12, "Referrals," on page 155.
- **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 10, “Alias,” on page 137.

- **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 backend, to the LDAP server schema, or to a RACF user, group, or connection profile. For more information, see Chapter 11, “Change logging,” on page 145.
- **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 communications**. 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 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 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 Version 3 protocol support**. The LDAP server provides support for Version 3 of the LDAP protocol in addition to the LDAP Version 2 protocol. Version 3 includes:
  - All protocol operations
  - Implicit bind
  - Certificate (or Simple Authentication and Security Layer) bind
  - Version 3 referrals
  - Aliases
  - Controls
  - Root DSE support
  - 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 Version 3 protocol. 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 communications. In order to use a certificate on bind, the server must be configured to perform both client and server authentication. This ensures

both entities are who they claim to be. 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 6, “CRAM-MD5 and DIGEST-MD5 authentication,” on page 81.
- **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 177.
- **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) 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
  IBMModifyDNRealignDNAttributesControl
  IBMModifyDNTimelimitControl
  IBMSchemaReplaceByValueControl
  manageDsaIT
  PersistentSearch
  replicateOperationalAttributes

  For more information, see Appendix B, “Supported server controls,” on page 219.
- **Attribute encryption**. The LDAP server supports encryption of the values of several critical attributes to prevent unauthorized access to these attribute values in LDBM backends. The attributes that can be encrypted are as follows:
  replicaCredentials
  secretKey
  userPassword

  For more information, see “Configuring for Encryption” 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 or the IPv4 or IPv6 address, 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 219.
- **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-allMembers** and **ibm-allGroups**. The LDAP server now supports the querying of the members of static, dynamic, and nested groups in an LDBM 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*.

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 (usually some sort of name), and continue with progressively broader attributes, often ending with a country attribute.

Relative distinguished names

Each component of a DN is referred to as a *relative distinguished name (RDN)*. It identifies an entry distinctly from any other entries which have the same parent. In the examples above, the RDN `cn=Ben Gray` separates the first entry from the second entry, (with RDN `cn=Lucille White`). The attribute=value pair or pairs making up the RDN for an entry must also be present as an attribute=value pair or pairs in the entry. This is not true of the other components of the DN. When using the LDBM backend, LDBM adds the attribute=value pairs in the RDN to the entry if they are not already present.

RDNs can contain multiple attribute=value pairs. So-called multivalued RDNs use two or more attribute=value pairs from the directory entry to define the name of the entry relative to its parent. An example where this would be useful would be where a directory hierarchy of users was being defined for a large university. This hierarchy would be segmented by campus. A problem is encountered, however, when it is discovered that there is more than one John Smith at the downtown campus. The RDN cannot simply be the name of the user. What can be done, however, is to add a unique value to the RDN, 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.

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
```

Chapter 3. LDAP directory schema

The LDAP Version 3 (V3) protocol, as defined in IETF RFC 2252 *Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions* and IETF RFC 2256 *A Summary of the X.500(96) User Schema for use with LDAPv3*, describes schema publication and update. Schema publication provides the ability to query the active directory schema through the use of the LDAP search function. Schema update is the ability to change the schema while the directory server is running.

Note:

- The z/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.
- 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 administrator are subject to ACL checking. In particular for a 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 administrator can update the schema. This ACL can be modified. See Chapter 8, “Using access control” for more information.

Setting up the schema for LDBM - new users

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. These files are USRSCHEM LDIF and IBMSCHEM LDIF and are located on the TCPMAINT 591 disk. The IBMSCHEM LDIF schema definitions require that the definitions contained in USRSCHEM LDIF are loaded prior to loading IBMSCHEM LDIF. Determine which of these schema files would be used to represent the data to be stored in the LDBM database, or locate or create other schema files to use.

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

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

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

Upgrading schema for LDBM

The schema files that are shipped with the z/VM LDAP server are based on industry and product defined schemas. As such, they should not be modified since existing products and applications use the schema elements as defined.

Occasionally, schema updates are required during the life of an LDAP release. These updates are applied to and shipped with the USRSCHEM LDIF and

IBMSCHEM LDIF files found on the TCPMAINT 591 disk. When moving to a new release you must reapply both of these files if you previously applied these schema files. Future schema service will depend on those updates being applied to your schema.

If you are using the USRSCHEM LDIF and IBMSCHEM LDIF schema files and either the files are updated in the service stream or you are moving to a new release, the LDAP Administrator should update the LDAP server schema through the **ldapmodify** utility. Run the following **ldapmodify** commands to load the schema:

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

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

Notes:

1. Check that **schemaReplaceByValue off** is not specified in the global section of the LDAP server configuration file or send the **IBMSchemaReplaceByValueControl** control with a value of **TRUE** on the modify request. This control can be sent by specifying the **-u** option on the **ldapmodify** utility. Refer to "Configuring the LDAP Server" in *z/VM: TCP/IP Planning and Customization* for more information on the **schemaReplaceByValue** configuration option and to Appendix B, "Supported server controls" for more information on the **IBMSchemaReplaceByValueControl** control.
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, Table 5, and Table 6. 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 and Table 5.

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	-	-	-
Boolean	booleanMatch	-	-
Directory String	caseIgnoreMatch	caseIgnoreOrderingMatch	caseIgnoreSubstringsMatch
DIT Content Rule Description	objectIdentifierFirstComponentMatch	-	-
DIT Structure Rule Description	integerFirstComponentMatch	-	-
Distinguished Name	distinguishedNameMatch	distinguishedNameOrderingMatch	-
Generalized Time	generalizedTimeMatch	generalizedTimeOrderingMatch	-
IA5 String	caseIgnoreIA5Match	caseIgnoreOrderingMatch	caseIgnoreSubstringsMatch
IBM Attribute Type	objectIdentifierFirstComponentMatch	-	-
IBM Entry UUID	IBM-EntryUUIDMatch	-	-
Integer	integerMatch	-	-
LDAP Syntax Description	objectIdentifierFirstComponentMatch	-	-
Matching Rule Description	objectIdentifierFirstComponentMatch	-	-
Matching Rule Use Description	objectIdentifierFirstComponentMatch	-	-
Name Form Description	objectIdentifierFirstComponentMatch	-	-
Object Class Description	objectIdentifierFirstComponentMatch	-	-
Object Identifier	objectIdentifierMatch	-	-
Octet String	octetStringMatch	-	-
Substring Assertion	-	-	-
Telephone Number	telephoneNumberMatch	-	telephoneNumberSubstringsMatch

Table 1. Syntax and default EQUALITY, ORDERING, and SUBSTR matching rules (continued)

Syntax	EQUALITY	ORDERING	SUBSTR
UTC Time	utcTimeMatch	-	-

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 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	-	-	-
Boolean	booleanMatch caseIgnoreMatch caseExactMatch	-	-
Directory String	caseIgnoreMatch caseExactMatch	caseIgnoreOrderingMatch caseExactOrderingMatch	caseIgnoreSubstringsMatch caseExactSubstringsMatch
DIT Content Rule Description	objectIdentifierFirstComponentMatch	-	-
DIT Structure Rule Description	integerFirstComponentMatch	-	-
Distinguished Name	distinguishedNameMatch	distinguishedNameOrdering Match	-
Generalized Time	generalizedTimeMatch	generalizedTimeOrdering Match	-
IA5	caseIgnoreMatch caseIgnoreIA5Match caseExactMatch caseExactIA5Match	caseIgnoreOrderingMatch caseExactOrderingMatch	caseIgnoreSubstringsMatch caseExactSubstringsMatch
IBM Attribute Type Description	objectIdentifierFirstComponent Match	-	-
IBM Entry UUID	IBM-EntryUUIDMatch	-	-
Integer	integerMatch integerFirstComponentMatch	-	-
LDAP Syntax Description	objectIdentifierFirstComponentMatch	-	-
Matching Rule Description	objectIdentifierFirstComponentMatch	-	-
Matching Rule Use Description	objectIdentifierFirstComponentMatch	-	-
Name Form Description	objectIdentifierFirstComponentMatch	-	-
Object Class Description	objectIdentifierFirstComponentMatch	-	-
Object Identifier	objectIdentifierMatch objectIdentifierFirstComponentMatch	-	-
Octet String	octetStringMatch	-	-
Substring Assertion	-	-	-
Telephone 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. 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 9, "Replication" for more information.

Note: The LDAP V3 protocol also defines a **COLLECTIVE** key word for attribute types. The LDAP server does not support this key word. All attribute types are assumed to be not **COLLECTIVE**.

- **IBM attribute types**

  Additional information required by IBM LDAP servers for each attribute type defined in the schema is specified using the **IBMAttributeTypes** schema attribute. The **IBMAttributeTypes** schema attribute is an extension of the **attributeTypes** schema attribute. If the **attributeTypes** value is not defined, then the corresponding **IBMAttributeTypes** value cannot be defined. For the z/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 100 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 Figure 2. The schema entry specifies **top**, **subEntry**, **subSchema**, and **ibmSubschema** as object classes. The object classes form the following hierarchy:

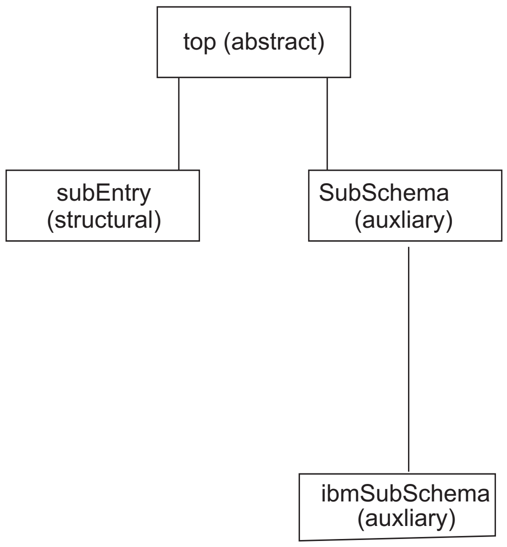


Figure 3. Object class hierarchy example

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 and Table 5 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 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. 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 )`
oids	Either 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 the LDAP V3 protocol 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.

Table 3. Character representations (continued)

Term	**Definition**
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 only.

The format of the LDAP syntaxes attribute in a dynamic schema is:

```
ldapSyntaxes: ( numericoid  [DESC qdstring] )
```

numericoid
: The unique, assigned numeric object identifier.

DESC *qdstring*
: Text description of the LDAP syntax

Note: LDAP syntaxes do not have a textual name. They are identified only by the numeric object identifier.

Following is an example of the definition of an LDAP syntax:

```
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.7 DESC 'Boolean' )
```

The LDAP syntaxes supported by the z/VM LDAP server fall into two categories. The first set, as shown in Table 4, 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
1.3.6.1.4.1.1466.115.121.1.7	Boolean	TRUE, FALSE
1.3.6.1.4.1.1466.115.121.1.15	Directory String	UTF-8 characters
1.3.6.1.4.1.1466.115.121.1.12	Distinguished Name	Sequence of attribute type and value pairs

Table 4. Supported LDAP syntaxes - general use (continued)

Numeric object identifier	Description	Valid values
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 will default to 0.
1.3.6.1.4.1.1466.115.121.1.26	IA5 String	IA5 characters (commonly known as 7-bit ASCII)
1.3.6.1.4.1.1466.115.121.1.27	Integer	+/- 62 digit integer
1.3.6.1.4.1.1466.115.121.1.38	Object Identifier	Name or numeric object identifier
1.3.6.1.4.1.1466.115.121.1.40	Octet String	Octet data
1.3.6.1.4.1.1466.115.121.1.50	Telephone Number	printable string (alphabetic, decimal, ″, **(**, **)**, **+**, **,**, **-**, **.**, **/**, **:**, **?**, and space)
1.3.6.1.4.1.1466.115.121.1.53	UTC Time	See Generalized Time above for details

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 only.

The format of the matching rules attribute in a dynamic schema is:

```
matchingRules: ( numericoid [NAME qdescrs] [DESC qdstring] [OBSOLETE] SYNTAX numericoid )
```

numericoid
: The unique, assigned numeric object identifier.

NAME *qdescrs*
: The name by which this matching rule is known.

DESC *qdstring*
: Text description of the matching rule.

OBSOLETE
: Indicates that the matching rule is obsolete.

SYNTAX *numericoid*
: Specifies the numeric object identifier of the syntax for this matching rule.

Following is an example of the definition of a matching rule:

```
matchingRules: ( 2.5.13.5 NAME 'caseExactMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

The matching rules supported by the z/VM LDAP server is a fixed set as listed in the following table.

Table 6. Supported matching rules

Name	Numeric object identifier	Assertion syntax
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.
caseIgnoreMatch	2.5.13.2	Directory String. 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
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.
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 will be normalized as yyyymmddhhmmss.ffffffZ.
generalizedTimeOrderingMatch	2.5.13.28	Generalized Time. The value will be 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.
objectIdentifierMatch	2.5.13.0	Object Identifier. The value will be normalized as an attribute descriptor.
objectIdentifierFirstComponentMatch	2.5.13.30	Object Identifier. The value will be 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.
telephoneNumberMatch	2.5.13.20	Telephone Number
telephoneNumberSubstringsMatch	2.5.13.21	Telephone Number. The value will be normalized using the telephoneNumberMatch rule.
utcTimeMatch	2.5.13.25	UTC Time. The value will be 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. The assertion value is just the DN portion of the attribute value.
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 backend ignores the **ORDERING** and **SUBSTR** matching rules and always uses the **EQUALITY** matching rule when processing a search filter.

Attribute types

The format of the attribute types attribute in a dynamic schema is:

```
attributeTypes: ( numericoid  [NAME qdescrs] [DESC qdstring] [OBSOLETE] [SUP oid]
 [EQUALITY oid] [ORDERING oid] [SUBSTR oid] [SYNTAX noidlen] [SINGLE-VALUE]
 [NO-USER-MODIFICATION] [USAGE attributeUsage] )
```

numericoid
: The unique, assigned numeric object identifier.

NAME *qdescrs*
: The name and alias names by which this attribute type is known. This is also known as the object identifier. The first name in the list is used as the base name and the other names are referred to as alias names. It is suggested the shortest name be listed first. If a name is not specified, the numeric object identifier is used to refer to the attribute type.

DESC *qdstring*
: Text description of the attribute type.

OBSOLETE
: Indicates that the attribute type is obsolete.

SUP *oid*
: Specifies the superior attribute type. When a superior attribute type is defined, the **EQUALITY**, **ORDERING**, **SUBSTR**, and **SYNTAX** values 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. 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 for the definition of these values.

The **IBMAttributeTypes** schema element is an extension of the **attributeTypes** schema element. If the **attributeTypes** value is not defined, then the corresponding **IBMAttributeTypes** value cannot be defined.

Some schema elements are shipped with **ACCESS-CLASS** set to **restricted** or **system**. These values are 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 Internet Assigned Numbers Authority (IANA) and, can be found at:

```
http://www.iana.org
```

Select the “Application Forms” link and then the “Private Enterprise Number” link to apply for a Private Enterprise number.

Once you have obtained an “OID arc” you can begin assigning OIDs to object classes and attribute types that you define.

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

```
dn: cn=schema
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 and GDBM backends, but will need to be updated for usage of LDBM. The schema files shipped with the LDAP server, USRSCHEM LDIF and IBMSCHEM LDIF, might be sufficient for your usage of LDBM. (For more information on adding these files to the schema, see "Setting up the schema for LDBM - new users" 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 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 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

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" 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	(can only move an attribute to **MAY** if the object class is in use by an entry)
MAY	(can only add an attribute if the object class is in use by an entry)

The **MUST** and **MAY** lists can be modified in any way if no directory entries are using this object class.

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.

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. Refer to “Configuring the LDAP Server” in *z/VM: TCP/IP Planning and Customization* for more information.
2. Specify the **IBMschemaReplaceByValueControl** control on the modify with replace operation to set the behavior for just that specific modify operation, overriding the **schemaReplaceByValue** configuration option. Specifying **TRUE** in the control activates the schema-replace-by-value behavior; **FALSE** activates the standard RFC behavior. Refer to Appendix B, “Supported server controls” 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. 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.

```
ldapsearch -h ldaphost -p ldapport -s base -b "cn=schema" "objectclass=subschema"
```

Immediately after the server is started for the first time, this command produces the results shown in Appendix A, "Initial LDAP server schema." 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.

```
ldapsearch -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 only supported in the LDBM (file-based) backend.

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 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"):

- **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 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 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 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 **ldapmodrdn**. Refer to the *z/VM: TCP/IP User's Guide* for more information on the **ldapmodrdn** utility.

Example 1: Simple Modify DN of leaf node

```
ModifyDNRequest ::= {
entry           cn=Kevin Heard, o=Athletics, o=Human Resources, o=Deltawing, c=AU
newrdn          cn=Kevin T. Heard
deleteoldrdn    TRUE
}


ldapmodrdn -h ldaphost -p ldapport -D binddn -w passwd -r "cn=Kevin Heard,
  o=Athletics, o=Human Resources, o=Deltawing, c=AU" "cn=Kevin T. Heard"
```

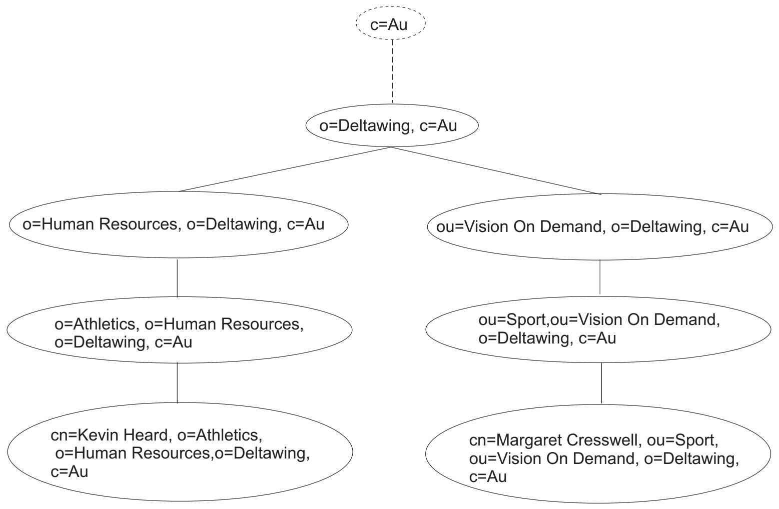

c=Au
o=Deltawing, c=Au
o=Human Resources, o=Deltawing, c=Au
ou=Vision On Demand, o=Deltawing, c=Au
o=Athletics, o=Human Resources, o=Deltawing, c=Au
ou=Sport,ou=Vision On Demand, o=Deltawing, c=Au
cn=Kevin Heard, o=Athletics, o=Human Resources,o=Deltawing, c=Au
cn=Margaret Cresswell, ou=Sport, ou=Vision On Demand, o=Deltawing, c=Au

Figure 4. Before Modify DN operation

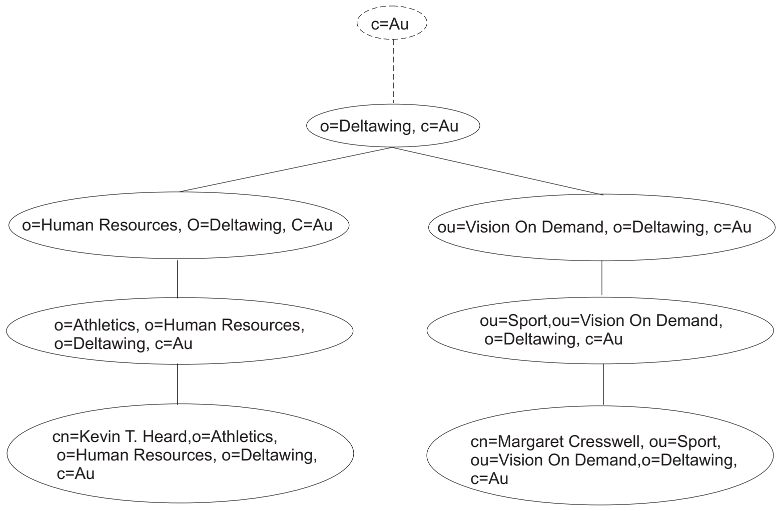

c=Au
o=Deltawing, c=Au
o=Human Resources, O=Deltawing, C=Au
ou=Vision On Demand, o=Deltawing, c=Au
o=Athletics, o=Human Resources, o=Deltawing, c=Au
ou=Sport,ou=Vision On Demand, o=Deltawing, c=Au
cn=Kevin T. Heard,o=Athletics, o=Human Resources, o=Deltawing, c=Au
cn=Margaret Cresswell, ou=Sport, ou=Vision On Demand,o=Deltawing, c=Au

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
 }
```

```
ldapmodrdn -h ldaphost -p ldapport -D binddn -w passwd "o=Athletics,
 o=Human Resources, o=Deltawing, c=AU" "ou=College Athletics Dept."
```

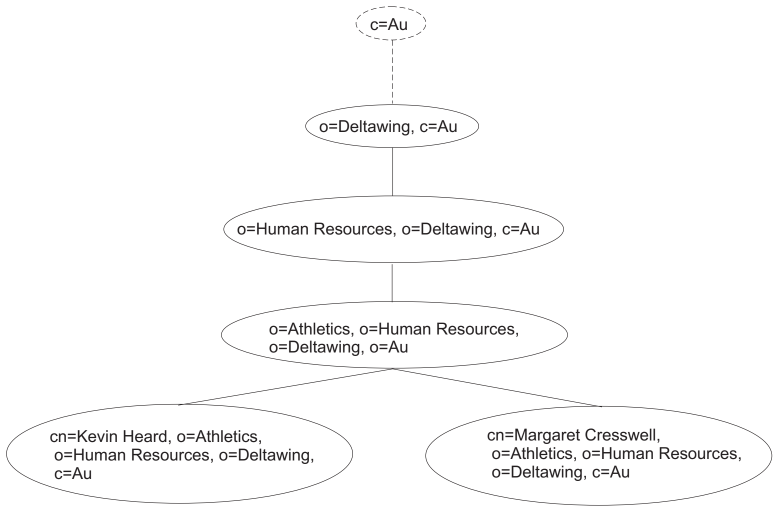

c=Au
o=Deltawing, c=Au
o=Human Resources, o=Deltawing, c=Au
o=Athletics, o=Human Resources, o=Deltawing, o=Au
cn=Kevin Heard, o=Athletics, o=Human Resources, o=Deltawing, c=Au
cn=Margaret Cresswell, o=Athletics, o=Human Resources, o=Deltawing, c=Au

Figure 6. Before Modify DN operation

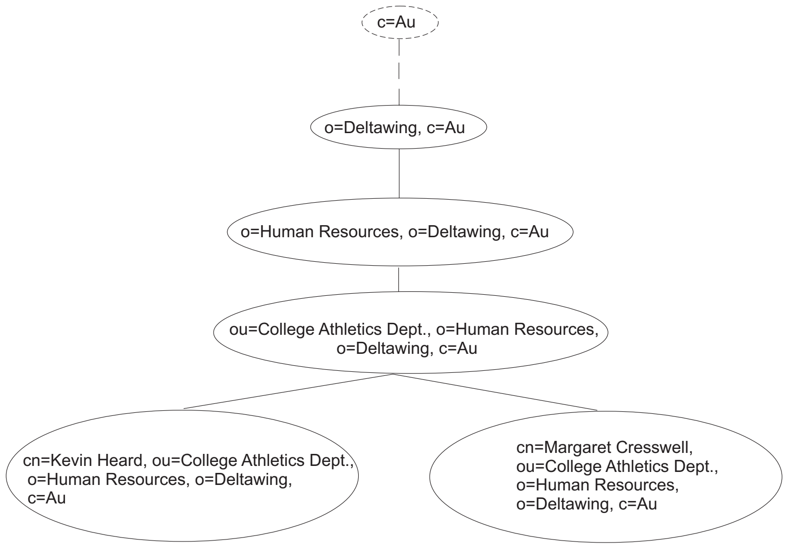

c=Au
o=Deltawing, c=Au
o=Human Resources, o=Deltawing, c=Au
ou=College Athletics Dept., o=Human Resources, o=Deltawing, c=Au
cn=Kevin Heard, ou=College Athletics Dept., o=Human Resources, o=Deltawing, c=Au
cn=Margaret Cresswell, ou=College Athletics Dept., o=Human Resources, o=Deltawing, c=Au

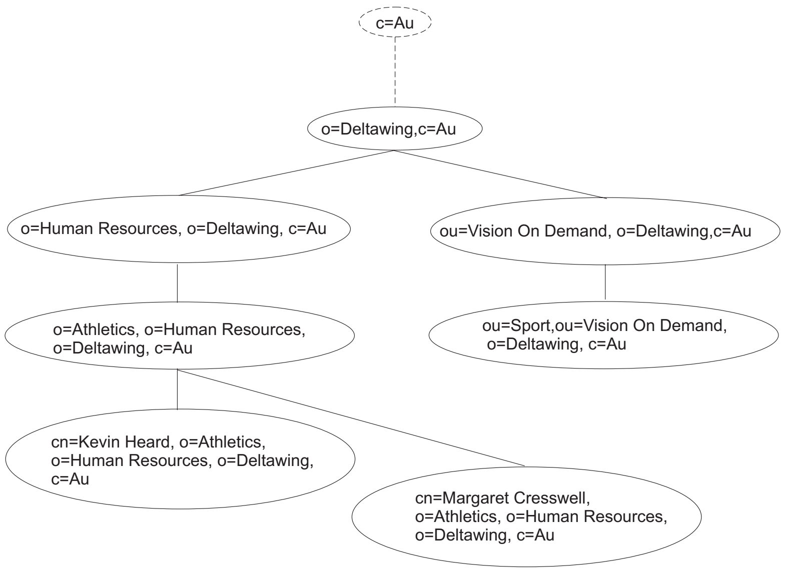
Figure 7. After Modify DN operation

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*)

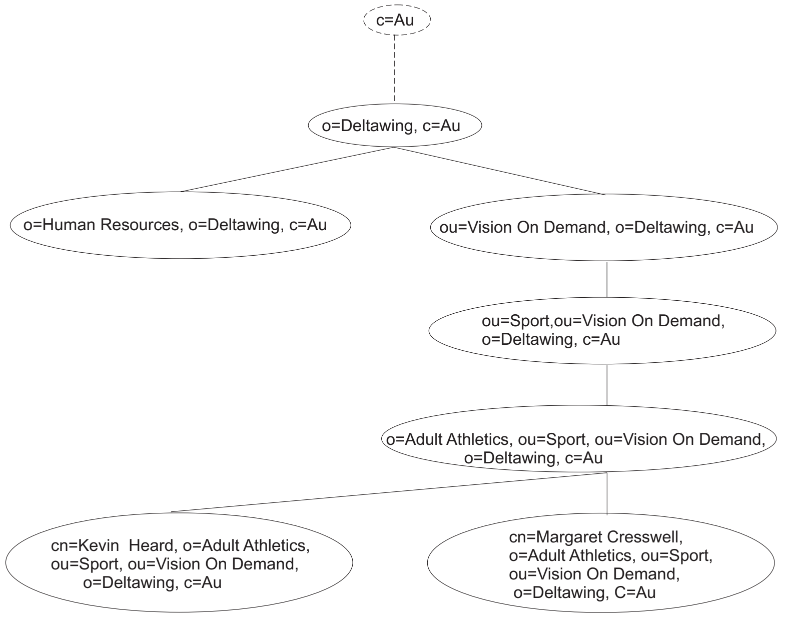
```
ModifyDNRequest ::= {
entry           o=Athletics, o=Human Resources, o=Deltawing, c=AU
newrdn          o=Adult Athletics
deleteoldrdn    FALSE,
newSuperior     ou=Sport, ou=Vision On Demand, o=Deltawing, o=AU
                }
```

```
ldapmodrdn -h ldaphost -p ldapport -D binddn -w passwd -s "ou=Sport, ou=Vision On
 Demand, o=Deltawing, c=AU" "o=Athletics,o=Human Resources, o=Deltawing, c=AU"
 "o=Adult Athletics"
```



c=Au
o=Deltawing,c=Au
o=Human Resources, o=Deltawing, c=Au
ou=Vision On Demand, o=Deltawing,c=Au
o=Athletics, o=Human Resources, o=Deltawing, c=Au
ou=Sport,ou=Vision On Demand, o=Deltawing, c=Au
cn=Kevin Heard, o=Athletics, o=Human Resources, o=Deltawing, c=Au
cn=Margaret Cresswell, o=Athletics, o=Human Resources, o=Deltawing, c=Au

Figure 8. Before Modify DN operation



c=Au
o=Deltawing, c=Au
o=Human Resources, o=Deltawing, c=Au
ou=Vision On Demand, o=Deltawing, c=Au
ou=Sport,ou=Vision On Demand, o=Deltawing, c=Au
o=Adult Athletics, ou=Sport, ou=Vision On Demand, o=Deltawing, c=Au
cn=Kevin Heard, o=Adult Athletics, ou=Sport, ou=Vision On Demand, o=Deltawing, c=Au
cn=Margaret Cresswell, o=Adult Athletics, ou=Sport, ou=Vision On Demand, o=Deltawing, C=Au

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 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 for more information.
- There are limitations to which directory entries are eligible for the Modify DN operation. See Eligibility of entries for rename for more information.
- In case the directory needs to be returned to a state prior to a Modify DN operation, the directory should be backed up by using the **ds2ldif** utility program. For more information about the **ds2ldif** utility program, 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 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 backend targeted by the Modify DN operation. The Modify DN operation with *newSuperior* option will move subtree entries within the same LDBM 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 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.

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 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: reportingOrganization	**access-class:** sensitive
attribute: workingOrganization	**access-class:** normal

The LDIF format representation of the entries containing **reportingOrganization** or **workingOrganization** attributes are:

```
dn: cn=Lisa Fare, ou=Human Resources, ou=Delta Home Media Ltd., o=Deltawing, c=AU
cn: Lisa Fare
objectclass: organizationalPerson
objectclass: person
objectclass: TOP
aclEntry: access-id: cn=Mark Crawford, ou=Human Resources, ou=Delta Home
Media Ltd., o=Deltawing,c=AU:normal:rsc:sensitive:rs
sn: Fare
title: Occupational Health and Safety Administrator
telephonenumber: (07) 635 1432
manager: cn=John Gardner, ou=Human Resources Group, ou=Deltawing InfoSystems,
 o=Deltawing, c=AU
secretary: cn=Ian Campbell, o=Deltawing, c=AU
reportingOrganization: o=Athletics, o=Human Resources, ou=Delta Home Media Ltd.,
 o=Deltawing, c=AU

dn: cn=Laurie Wood, ou=Human Resources Group, ou=Deltawing Automotive Ltd., o=Deltawing, c=AU
cn: Laurie Wood
objectclass: organizationalPerson
objectclass: person
objectclass: TOP
aclEntry: access-id: cn=Mark Crawford, ou=Human Resources, ou=Delta Home
 Media Ltd., o=Deltawing,c=AU:normal:rswc:sensitive:rsw
sn: Wood
telephonenumber: (03) 9335 2114
title: Pay Officer
workingOrganization: o=Athletics, o=Human Resources, ou=Delta Home Media Ltd.,
 o=Deltawing, c=AU
```

Relocating an entry

User "cn=Mark Crawford, ou=Human Resources, ou=Delta Home Media Ltd., o=Deltawing,c=AU" submits the following Modify DN operation request to the server to relocate the target entry:

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

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

Relocating an entry with DN realignment requested

If the same user submits a Modify DN operation request to the server to relocate the same target entry under the same newSuperior entry, but with the addition of the control requesting realignment of DN attribute values (**-a** parameter):

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

In addition to the permissions required on the previous example, this operation requires additional permissions to be checked on entries containing values which qualify for realignment. The DN being modified ("o=Athletics, o=Human Resources, ou=Delta Home Media Ltd., o=Deltawing, c=AU") is found in DN-syntax attributes of two entries: The entry with DN "cn=Laurie Wood, ou=Human Resources Group, ou=Deltawing Automotive Ltd., o=Deltawing, c=AU" contains this value in the **workingOrganization** attribute, and the entry with DN "cn=Lisa Fare, ou=Human Resources, ou=Delta Home Media Ltd., o=Deltawing, c=AU" contains this value in the **reportingOrganization** attribute.

The bound user is in the aclEntry for "cn=Laurie Wood, ou=Human Resources Group, ou=Deltawing Automotive Ltd., o=Deltawing, c=AU". The **workingOrganization** attribute is in the access-class of normal, and the bound user is granted **w** access to this class of attributes, so the realignment of the DN value would be permitted in this entry.

The bound user is also in the **aclEntry** for "cn=Lisa Fare, ou=Human Resources, ou=Delta Home Media Ltd., o=Deltawing, c=AU". The **reportingOrganization** attribute is in the access-class of sensitive, and the bound user is granted only **rs** permissions on **sensitive** attributes in the entry, so the realignment of this value would be denied. Even though the bound user had adequate permissions to perform the relocation of the target entry and had adequate permissions to perform realignment of the DN value in one of the two entries containing a matching DN, the operation would fail because the bound user does not have the necessary permissions on everything 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, prior to such a move or rename users should carefully consider how ownership and accessibility to entries protected by these attributes may change after the move, and what ACL and ownership changes may be desired, if any.

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

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

```
ldapmodrdn -h ldaphost -p ldapport -D binddn -w passwd -s "ou=Sport,
 ou=Vision On Demand, o=Deltawing, c=AU" "o=Athletics,o=Human Resources,
 o=Deltawing, c=AU" "o=Adult Athletics"
```

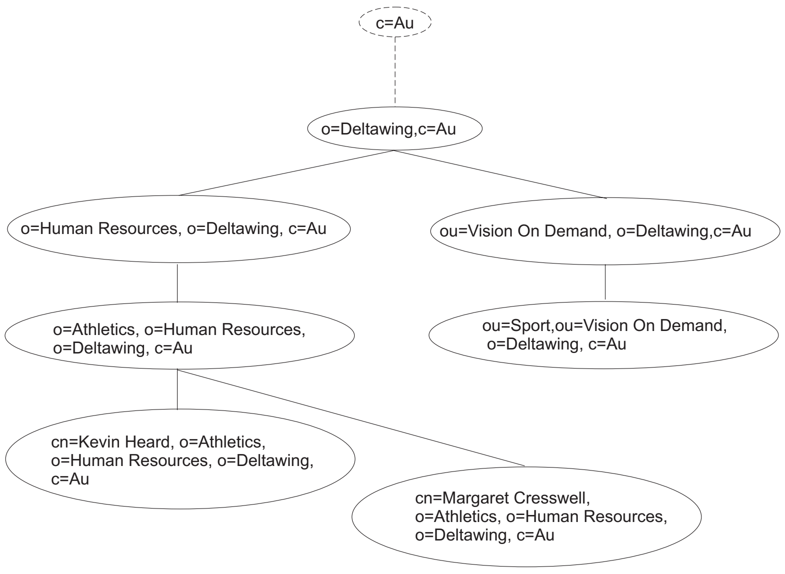

c=Au
o=Deltawing,c=Au
o=Human Resources, o=Deltawing, c=Au
ou=Vision On Demand, o=Deltawing,c=Au
o=Athletics, o=Human Resources, o=Deltawing, c=Au
ou=Sport,ou=Vision On Demand, o=Deltawing, c=Au
cn=Kevin Heard, o=Athletics, o=Human Resources, o=Deltawing, c=Au
cn=Margaret Cresswell, o=Athletics, o=Human Resources, o=Deltawing, c=Au

Figure 10. Before Modify Dn operation

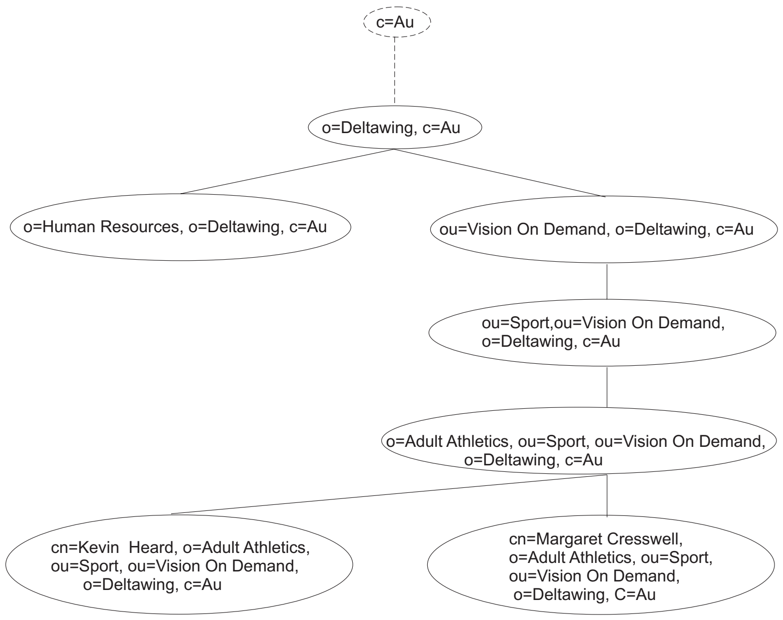

c=Au
o=Deltawing, c=Au
o=Human Resources, o=Deltawing, c=Au
ou=Vision On Demand, o=Deltawing, c=Au
ou=Sport,ou=Vision On Demand, o=Deltawing, c=Au
o=Adult Athletics, ou=Sport, ou=Vision On Demand, o=Deltawing, c=Au
cn=Kevin Heard, o=Adult Athletics, ou=Sport, ou=Vision On Demand, o=Deltawing, c=Au
cn=Margaret Cresswell, o=Adult Athletics, ou=Sport, ou=Vision On Demand, o=Deltawing, C=Au

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 wish to return the moved entry (and its subordinates) back to their original location in the directory hierarchy, this will not be possible with the access controls currently in place. The bound DN has only **object:d** permission on the old superior node ("o=Human Resources, o=Deltawing, c=AU") where **object:a** is needed to effect the move of an entry or subtree under the superior node, and the bound DN has only **object:a** permission on the moved entry ("o=Adult Athletics, ou=Sport, ou=Vision On Demand, O=Deltawing, c=AU") where **object:d** is needed to move the entry. 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 prior to the relocation will preserve that explicit ownership after the relocation has been performed. Any entries in the relocated subtree which inherited ownership prior to relocation will continue to inherit ownership following relocation. If the owning entry prior to relocation was a node superior to the relocated entry, the owning entry will be the new superior entry. If the owning entry was an entry within the relocated subtree, the owning entry is preserved following the relocation.

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

Refer to the example in Access control changes.

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 may 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.

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:

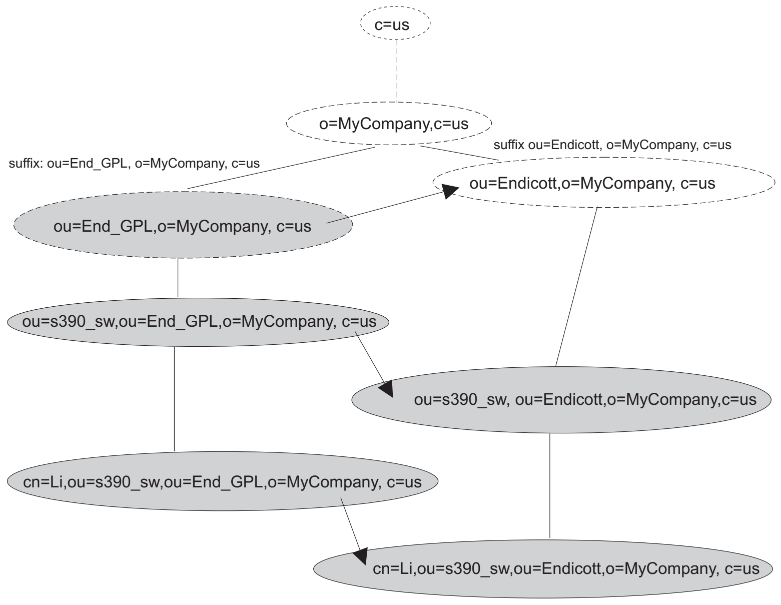

c=us
o=MyCompany,c=us
suffix ou=Endicott, o=MyCompany, c=us
suffix: ou=End_GPL, o=MyCompany, c=us
ou=Endicott,o=MyCompany, c=us
ou=End_GPL,o=MyCompany, c=us
ou=s390_sw,ou=End_GPL,o=MyCompany, c=us
ou=s390_sw, ou=Endicott,o=MyCompany,c=us
cn=Li,ou=s390_sw,ou=End_GPL,o=MyCompany, c=us
cn=Li,ou=s390_sw,ou=Endicott,o=MyCompany, 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:

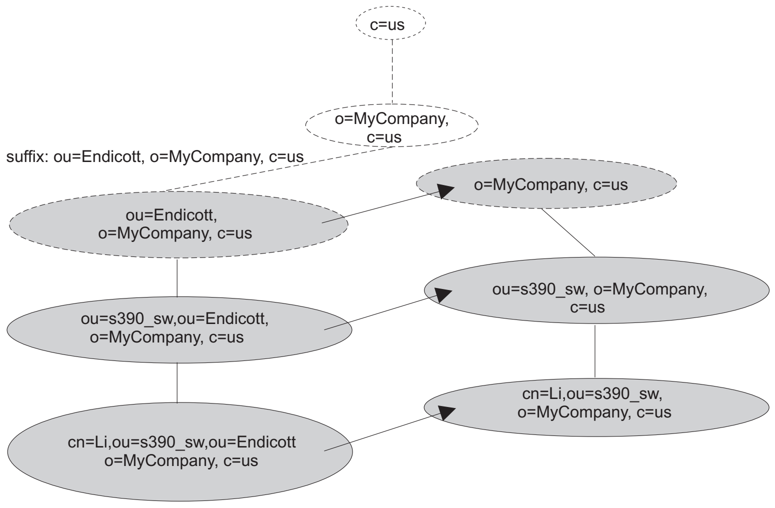

c=us
o=MyCompany, c=us
suffix: ou=Endicott, o=MyCompany, c=us
o=MyCompany, c=us
ou=Endicott, o=MyCompany, c=us
ou=s390_sw, o=MyCompany, c=us
ou=s390_sw,ou=Endicott, o=MyCompany, c=us
cn=Li,ou=s390_sw, o=MyCompany, c=us
cn=Li,ou=s390_sw,ou=Endicott o=MyCompany, c=us

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:

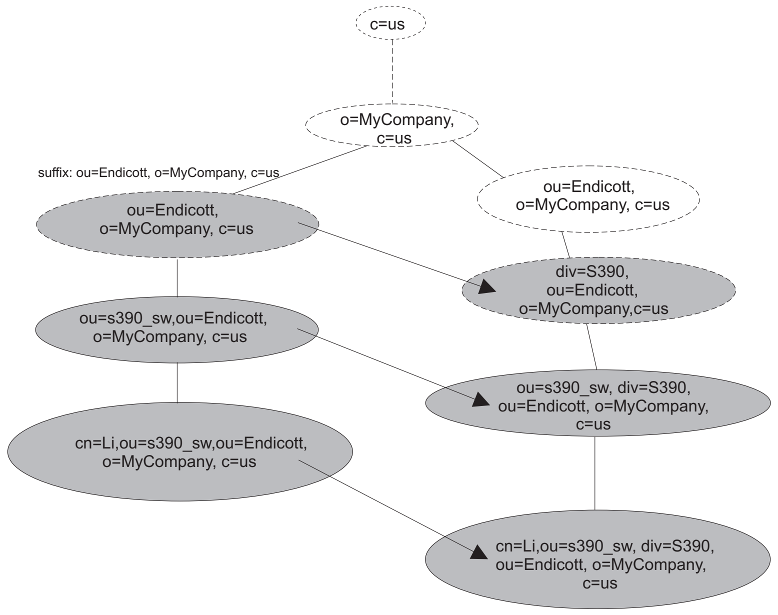

c=us
o=MyCompany, c=us
suffix: ou=Endicott, o=MyCompany, c=us
ou=Endicott, o=MyCompany, c=us
ou=Endicott, o=MyCompany, c=us
div=S390, ou=Endicott, o=MyCompany,c=us
ou=s390_sw,ou=Endicott, o=MyCompany, c=us
ou=s390_sw, div=S390, ou=Endicott, o=MyCompany, c=us
cn=Li,ou=s390_sw,ou=Endicott, o=MyCompany, c=us
cn=Li,ou=s390_sw, div=S390, ou=Endicott, o=MyCompany, c=us

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`

At this point, it should be noted that if these operational scenarios are to be replicated from a master server to one or more replica servers, there is a procedure which must be followed to permit this.

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

b. On the master server, perform the previous Steps 3a and 3b from the examples above. This will result in the intermediate rename to be performed on the master server and the results to be propagated to the replica server(s).

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

d. On the master server, perform the previous Step 3c from the examples above. This will result in the rename of entries to the final destination on the master server and in the results being propagated to the replica server(s).

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

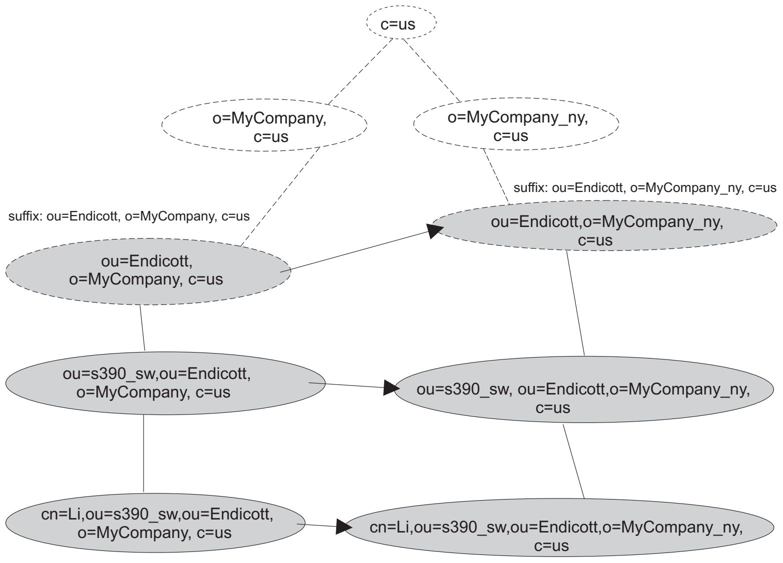

c=us
o=MyCompany, c=us
o=MyCompany_ny, c=us
suffix: ou=Endicott, o=MyCompany_ny, c=us
suffix: ou=Endicott, o=MyCompany, c=us
ou=Endicott,o=MyCompany_ny, c=us
ou=Endicott, o=MyCompany, c=us
ou=s390_sw,ou=Endicott, o=MyCompany, c=us
ou=s390_sw, ou=Endicott,o=MyCompany_ny, c=us
cn=Li,ou=s390_sw,ou=Endicott, o=MyCompany, c=us
cn=Li,ou=s390_sw,ou=Endicott,o=MyCompany_ny, c=us

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:

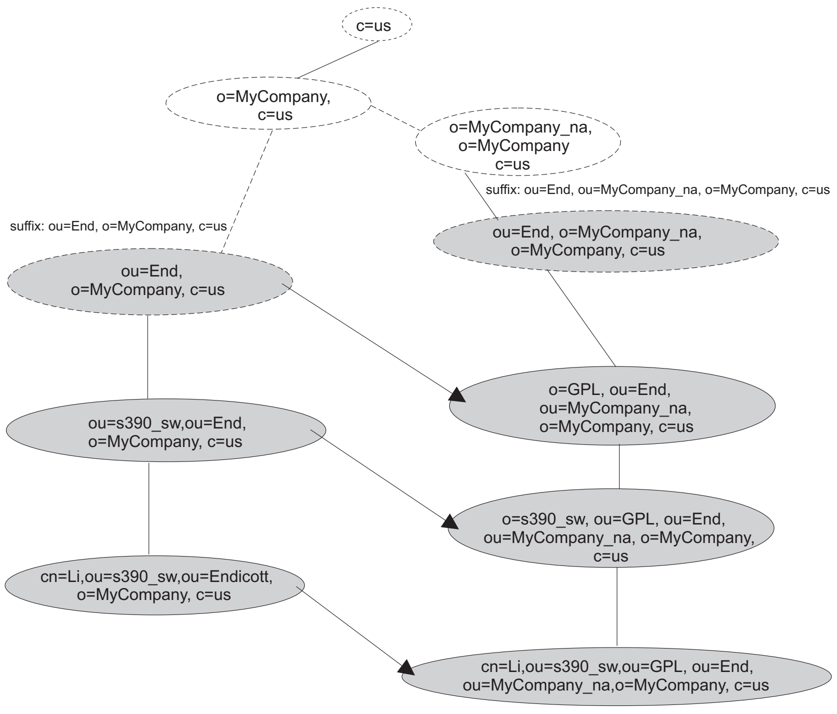

c=us
o=MyCompany, c=us
o=MyCompany_na, o=MyCompany c=us
suffix: ou=End, ou=MyCompany_na, o=MyCompany, c=us
suffix: ou=End, o=MyCompany, c=us
ou=End, o=MyCompany_na, o=MyCompany, c=us
ou=End, o=MyCompany, c=us
o=GPL, ou=End, ou=MyCompany_na, o=MyCompany, c=us
ou=s390_sw,ou=End, o=MyCompany, c=us
o=s390_sw, ou=GPL, ou=End, ou=MyCompany_na, o=MyCompany, c=us
cn=Li,ou=s390_sw,ou=Endicott, o=MyCompany, c=us
cn=Li,ou=s390_sw,ou=GPL, ou=End, ou=MyCompany_na,o=MyCompany, c=us

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:
   The new DN must be already designated as a suffix for this backend, otherwise

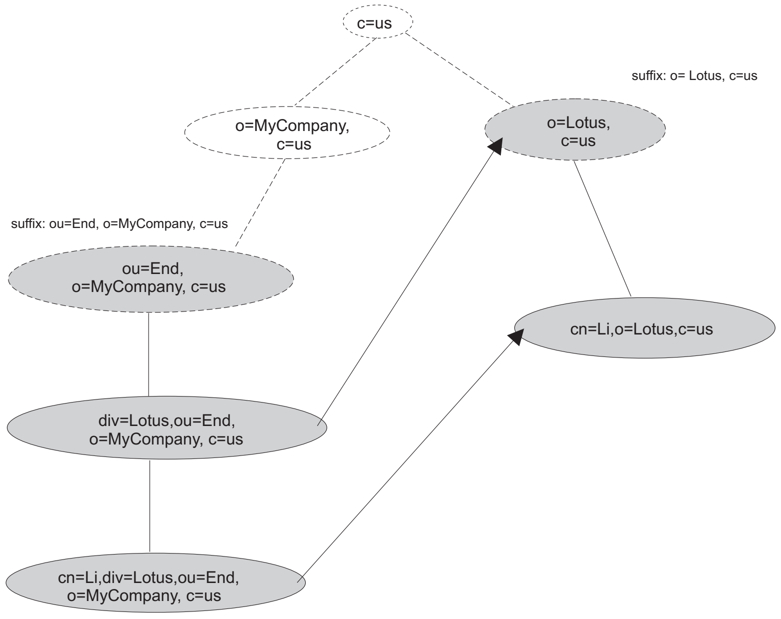

c=us
suffix: o= Lotus, c=us
o=MyCompany, c=us
o=Lotus, c=us
suffix: ou=End, o=MyCompany, c=us
ou=End, o=MyCompany, c=us
cn=Li,o=Lotus,c=us
div=Lotus,ou=End, o=MyCompany, c=us
cn=Li,div=Lotus,ou=End, o=MyCompany, c=us

Figure 17. Rename non-suffix entry to suffix entry

   this operation will fail.

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

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

Modify DN operations and replication

Modify DN operations may be classified into two categories:

1. Simple Modify DN operations are those which rename a leaf node, and which are not accompanied by the *newSuperior* parameter or the **IBMModifyDNRealignDNAttributesControl** control or the **IBMModifyDNTimelimitControl** control.

2. Complex Modify DN operations are those which either rename a mid-tree (non-leaf) node, or which are accompanied by the *newSuperior* parameter, or which are accompanied by either the **IBMModifyDNRealignDNAttributesControl** control or the **IBMModifyDNTimelimitControl** control.

Simple Modify DN operations are always accepted by the master server, and are replicated if replica entries are present in the LDBM backend where a Modify DN operation is applied.

Periodic validation of compatible server versions in replica servers

Periodic checks are made of replica servers by the master server which are intended to increase the likelihood that complex Modify DN operations will be successfully replicated. Following is a description of the mechanisms used by master servers to do such checking.

The LDAP server must be able to establish a connection to each of the replica servers represented by replica entries in an LDBM backend. When the connection is established to a given replica server, the master server determines if the replica server is at a compatible server version based on a query of the root DSE on that server. If a connection cannot be established to a replica server, it is assumed that the server does not provide the requisite support for replication of Modify DN operations, and complex Modify DN operations are refused at the master server. If a connection is established to a replica server and it is determined that the replica is not at a compatible server version, complex Modify DN operations are refused at the master server. Note that replication of simple Modify DN operations is always permitted, and such operations are always performed at the master server.

The master server may enable or disable processing of complex Modify DN operations, depending on dynamically changing states of replica servers and of replica entries within the master server's 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 may be established successfully to all replica servers and if they are determined to be running a compatible server version will the master server resume accepting complex Modify DN operations.

Other possible events which may influence whether the master server accepts or refuses complex Modify DN operations involve:

- the addition of new replica entries
- deletion of existing replica entries
- modification of existing replica entries in the LDBM backend.

Each of these causes the master server to temporarily suspend processing of complex Modify DN operations, until the check of replica servers at the start of the next replication cycle, at which point the replica server version levels will be used to determine whether the master server resumes accepting complex Modify DN operations.

To determine whether a replica server is at a compatible version level, submit a root DSE search to that server, similar to the following. The **-D** and **-w** options only need to be specified if the replica server does not support anonymous binds.

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

where *ldaphost* represents the 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 replication synchronization due to incompatible replica server versions

The LDAP Server replication model runs periodically, rather than continuously, and the state of the replica is not checked until the start of each replication cycle. A complex Modify DN operation could be accepted or rejected based on inaccurate information about the state of a replica server between the start of two replication cycles. As a consequence, the replication process could stall and the synchronization between the master server and its replicas could be lost.

Attention

It is highly recommended that 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 replication synchronization due to incompatible replica server versions - recovery

If at some point a master server accepts a complex Modify DN operation which can not be replicated, there are several means of recovering from this situation. The best method of recovering from this situation is to ensure that all replica servers are reachable from the master server, and that all replica servers are running at a compatible version level (this may entail stopping some replica servers and restarting them at a compatible version level). Once this state has been reached, queued changes awaiting propagation to replica servers will drain from the queue at the master server and the replication process will resume normal operation.

An alternative is to delete the 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 one wishes to later restart the offending replica (such as, after it has been brought up to a compatible server version) it will be necessary to take a backup of the master server contents and restore those contents to the replica server before restarting it, to ensure the two directories are synchronized.

Chapter 5. Accessing RACF information

RACF provides definitions of users and groups, as well as access control for resources. The LDAP server can provide LDAP access to the user and group information stored in RACF.

Using SDBM, the RACF database backend of the LDAP server, you can:
- Add new users and groups to RACF
- Add users to groups (connections)
- Modify RACF information for users, groups, and connections
- Retrieve RACF information for users, groups, and connections
- Delete users and groups from RACF
- Remove users from groups (connections)
- Retrieve RACF user password and password phrase envelopes

The SDBM backend of the LDAP server implements portions of the **adduser, addgroup, altuser, altgroup, deluser, delgroup, connect, remove**, and **search** RACF commands.

Note that the SDBM backend only updates the default RACF on a given system. That is, the **AT** and **ONLYAT** clauses of the RACF commands, used to redirect RACF commands, are not exploited by SDBM.

See *z/VM: RACF Security Server Command Language Reference* for more information about the supported RACF commands.

See “Setting up for SDBM” in *z/VM: TCP/IP Planning and Customization*. for information on getting your LDAP server configured with SDBM.

SDBM authorization

SDBM operations can be performed after several different types of binds to the LDAP server. In each of these binds, the LDAP server associates a RACF user ID with the bound user. SDBM 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” for more information.
- LDBM 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 will succeed as long as the **__passwd()** call made by the LDAP server is successful. However, no group membership information will be available for the bound distinguished name if the security manager is not RACF.
- An LDAP simple bind made to a z/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()** will have 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**.

SDBM group gathering

After successfully authenticating to the LDAP server, a list is created of the groups to which the authenticated RACF user ID belongs. Only groups in which the user ID’s membership is active (has not been revoked) are included in the list. This group membership list is used in authorization checking when trying to access entries in directories on the LDAP server.

If the SDBM backend is to be used for authentication purposes only and group membership is not needed, consider having your clients use the **authenticateOnly** server control, to streamline bind processing. This control overrides any extended group membership searching and default group membership gathering and is supported for Version 3 clients. See Appendix B, “Supported server controls” for more information.

Note the **authenticateOnly** control is not necessary if there is no LDBM or GDBM 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, and connection profile 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), group (Table 9), and connection (Table 10) 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
User base	PHRASE	racfPassPhrase

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	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.

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
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
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

Table 9. Mapping of LDAP attribute names to RACF fields (group) (continued)

RACF segment name	RACF keyword in altgroup/addgroup/ listgrp	LDAP attribute name
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

Special usage of racfAttributes and racfConnectAttributes

The **racfAttributes** attribute is a multi-valued attribute that can be used to specify any single-word keywords that can be specified on a RACF **adduser** or **altuser** command. For example, **racfAttributes** can be used to add a RACF user entry with ’ADSP GRPACC NOPASSWORD’ or modify a RACF user entry with ’NOGRPACC SPECIAL NOEXPIRED RESUME NOOVM’ . Additional values, such as PASSWORD, can be returned in **racfAttributes** that are not returned by the **listuser**.

Similarly, **racfConnectAttributes** can be used to specify any single-word keywords that can be specified on a RACF **connect** command.

RACF namespace entries

When the SDBM backend is used to make RACF information accessible over the LDAP protocol, the top four entries in the hierarchy are reserved, read-only, and generated by the server. The purpose of these reserved entries is to enable a hierarchical representation of RACF users, groups, and connections. For example, the top four entries in Figure 18 are:

- `cn=RACFA,o=IBM,c=US` (suffixDN)
- `profileType=User,cn=RACFA,o=IBM,c=US`
- `profileType=Group,cn=RACFA,o=IBM,c=US`
- `profileType=Connect,cn=RACFA,o=IBM,c=US`

The value of the top DN is generated from the suffix line in the **DS CONF** file for the SDBM backend entry (see "Setting up for SDBM" in *z/VM: TCP/IP Planning and Customization*.).

Following is a high-level diagram of the RACF backend.

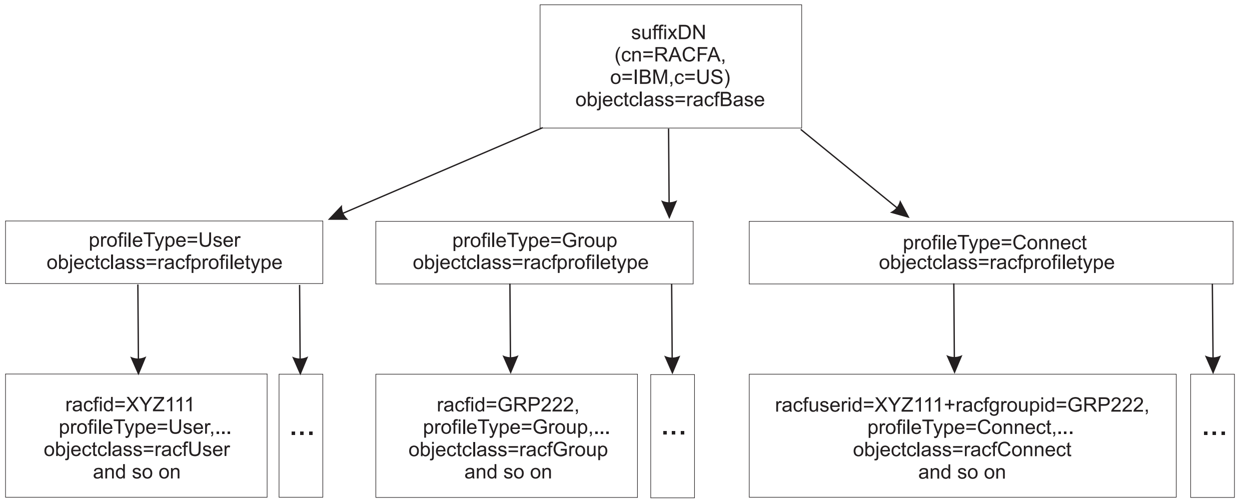


Figure 18. RACF namespace hierarchy

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 pound sign

An SDBM DN can contain a pound sign (#) anywhere in the DN, including the suffix. If the pound sign is at the beginning of a value in the DN, it 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 pound sign, but only a single backslash is used in a DN.

For example, if the SDBM suffix in the configuration file is

```
suffix cn=\\#plex#1
```

then the DN for the RACF user #abc# would be

```
racfid=\#abc#,profiletype=user,cn=\#plex#1
```

Pound signs at the beginning of a value in a DN returned by SDBM are always escaped by a single backslash. Other pound signs within the DN may or may not be escaped, depending on the usage of the DN.

When specifying a value starting with a pound sign for an attribute within an add or modify request, escape the pound sign with a back slash if the attribute is part of a DN, otherwise, do not escape the pound sign. For instance, to add a user with the default group `#d1grp`, specify either:

```
racfdefaultgroup: racfid=\#d1grp,profiletype=group,cn=\#plex#1
```

or

```
racfdefaultgroup: #d1grp
```

within the entry.

When specifying a value containing a pound sign for an attribute within a search filter, the pound sign 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 11 and Table 12 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 due to 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 11 shows how SDBM behaves during different LDAP operations.

Table 11. 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 **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 **Search subtree** See Searching the entire RACF database **Bind** Error: No credentials
profiletype=Group,suffixDN	**Add** Error: Unwilling to perform **Modify** Error: Unwilling to perform **Delete** Error: Unwilling to perform **Modify DN** Error: Unwilling to perform **Compare** Compare attribute **Search base** Return requested attributes **Search one level** See Searching the entire RACF database **Search subtree** See Searching the entire RACF database **Bind** Error: No credentials

Table 11. RACF backend behavior (continued)

Target DN	LDAP operation behavior
profiletype=Connect,suffixDN	**Add** Error: Unwilling to perform **Modify** Error: Unwilling to perform **Delete** Error: Unwilling to perform **Modify DN** Error: Unwilling to perform **Compare** Compare attribute **Search base** Return requested attributes **Search one level** See Searching the entire RACF database **Search subtree** See Searching the entire RACF database **Bind** Error: No credentials
racfid=XYZ111,profiletype=User, suffixDN	**Add** Perform an **adduser** RACF command using USER=XYZ111 **Modify** Perform an **altuser** RACF command using USER=XYZ111 **Delete** Perform a **deluser** RACF command using USER= XYZ111 **Modify DN** Error: Unwilling to perform **Compare** Compare requested attribute with data returned from 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

Table 11. RACF backend behavior (continued)

Target DN	LDAP operation behavior
racfid=GRP222,profiletype= Group, suffixDN	**Add** Perform an **addgroup** RACF command using GROUP=GRP222 **Modify** Perform an **altgroup** RACF command using GROUP=GRP222 **Delete** Perform a **delgroup** RACF command using GROUP=GRP222 **Modify DN** Error: Unwilling to perform **Compare** Compare requested attribute with data returned from 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
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

If LDAP is running with an SDBM backend, the **ldap_modify** and **ldap_add** APIs can return **LDAP_OTHER** or **LDAP_SUCCESS** and have completed a partial update to an entry in RACF. The results will match what would occur if the update were done using the RACF **altuser**, **altgroup**, and **connect** commands. If several RACF attributes are being updated and one of them is in error, RACF might still update the other attributes, without, in some cases, returning an error message. If there is a RACF message, LDAP always returns it in the result

The RACF **connect** command is used to both add a user connection to a group and to modify a user's connection to a group. As a result, the SDBM add and modify support for connection entries is different than normal LDAP support:

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

Notes about specifying attribute values:

1. There are several SDBM attributes whose value is a RACF user or group name. For convenience, this value can be specified either as just the RACF name or as the complete LDAP DN. For example, when adding a user with a default group of `grp222`, the **racfDefaultGroup** attribute can be specified as

   ```
   racfDefaultGroup: grp222
   ```

   or

   ```
   racfDefaultGroup: racfid=grp222,profiletype=group,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** command does not always support the ability to both add a value and replace the existing value. As a result, SDBM does not always respect the type of modification (add versus replace) that is specified in a modify command. Values for the following multi-value attributes are always added to the existing value (even if replace is specified): **racfAttributes**, **racfClassName**, **racfConnectAttributes**, **racfLevelKeyword**, **racfMformKeyword**, **racfMonitorKeyword**, **racfSecurityCategoryList**. Values for the following multi-value attributes always replace the existing value (even if add is specified): **racfDomains**, **racfMscopeSystems**, **racfNetviewOperatorClass**, **racfOperatorClass**, **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**.

   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. In order to update CICS-related attributes, CICS must be set up on your system; otherwise, errors result.
4. For modify, if a request is made to delete a specific attribute value for an attribute where specific values cannot be selectively deleted, LDAP_UNWILLING_TO_PERFORM is returned. There are four attributes where specific attribute values are accepted: **racfAttributes**, **racfClassName**, **racfConnectAttributes**, and **racfSecurityCategoryList**. If an attempt is made to delete any attribute that has no corresponding delete command in RACF, LDAP_UNWILLING_TO_PERFORM is returned.

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 the top four entries: the `suffix` entry, the `profiletype=user,suffix` entry, the `profiletype=group,suffix` entry, and the `profiletype=connect,suffix` entry.

Table 12. SDBM search filters

Filter	Search behavior
objectclass=*	**Description:** match any user, group, and connection profile **Allowed base:** any SDBM entry **Returns:** • DN-only entries if scope includes all users, groups, or connections • 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 a single user: **user profile extract** • if scope includes a single group: **group profile extract** • if scope includes a single connection: **connect profile extract**

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

Table 12. SDBM search filters (continued)

Filter	Search behavior

racfid=*any_value*	**Description:** find user and group profiles for the RACF users and groups whose names match *any_value* (can contain wildcards)
	
	Allowed base:
	`suffix`
	`profiletype=user,suffix`
	`profiletype=group,suffix`
	
	Returns: DN-only entries
	
	Commands:
	• if scope includes all users:
	search class(user) filter(*any_value*)
	• if scope includes all groups:
	search class(group) filter(*any_value*)

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

Table 12. 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:** `suffix` `profiletype=connect,suffix` **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 **racfid**, **racfuserid**, and **racfgroupid** filters can include the wild cards supported by RACF. These wild cards are '*' which represents any number of characters, and '%' which represents one character. For example:

```
(&(racfuserid=usr*)(racfgroupid=*grp))
```

searches for all the connections between users whose names begin with `usr` and groups whose names end with `grp`.

Note about searching universal groups: Most of the members of a RACF universal group are not actually contained in the group's list of members. As a result, a search of the entry for a universal group does not return most of the group's members. In addition, a search for the connection entry corresponding to a member of a universal group can return different results depending on the connection search filter that is used:

- If the **racfuserid** part of the connection search filter does not contain a wild card, then the connection entry is returned for the specified **racfuserid**.
- If the **racfuserid** part of the connection search filter contains a wild card, then the connection entry for a user is returned only if the user is explicitly contained in the universal group's list of members.

Searching the entire RACF database

Searches that query the entire RACF database, for example, a subtree search from one of the top four directory entries, return only the DN (distinguished name) attribute. You may then obtain more specific data about a particular user or group on a follow-up search using a specific DN as the search base.

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 may 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 may be truncated. In particular, make sure that the installation DATA field in the RACF user profile does not contain binary zeros or other unprintable characters.

Retrieving RACF user password and password phrase envelopes

SDBM returns the RACF user password envelope when the **racfPasswordEnvelope** attribute is specified in the attributes to be returned from a search of a RACF user. Similarly, the RACF user password phrase envelope is returned when the **racfPassPhraseEnvelope** attribute is specified on the search. Each envelope is returned by the LDAP server as a binary data berval (binary data and length). If the **racfPasswordEnvelope** and **racfPassPhraseEnvelope** attributes are not specified on the search request, the RACF envelopes are not returned.

Note: When using a utility such as **ldapsearch** to retrieve the password or password phrase envelopes, the returned value is base-64 encoded.

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`:

```
ldapsearch -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 **ldapmodify** to change the password. If the syntax of the new password is not valid, the command fails, returning "ldap_modify: Unknown error". (Note that this response can also be returned under other circumstances.)

   For example, the following LDIF file, `pw.mod`, resets the password for RACF user `U1` to `xyz`, assuming the SDBM suffix is `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:

```
ldapmodify -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 operation utilities with SDBM

The LDAP operation 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=ibm,c=us
objectclass: racfUser
racfid: newuser
```

The following command will add user ID `newuser` to RACF:

```
ldapadd -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,cn=racfu01,o=ibm,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=ibm,C=us
changetype: modify
objectclass: racfUserOvmSegment
racfOvmHome: /home/newuser
racfOvmInitialProgram: /home/newuser/bin/startup
racfOvmUid : 500
```

The command:

```
ldapmodify -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,
 cn=racfu01,o=ibm,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:

```
ldapsearch -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,
 cn=racfu01,o=ibm,c=us -w passwd -b "racfid=newuser,profiletype=user,cn=racfu01,o=ibm,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=IBM,c=US
racfid=NEWUSER
racfauthorizationdate=10/23/06
racfowner=RACFID=OPERATOR,PROFILETYPE=USER,CN=RACFU01,O=IBM,C=US
racfpasswordinterval=30
racfdefaultgroup=RACFID=SYS1,PROFILETYPE=GROUP,CN=RACFU01,O=IBM,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=IBM,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:

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

The result returned is:

```
dn: racfid=newuser,profiletype=user,cn=racfu01,o=ibm,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=ibm,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=ibm,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=ibm,c=us
objectclass: racfconnect
racfuserid: newuser
racfgroupid: grp222
```

The command:

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

```
ldapsearch -h ldaphost -p ldapport -D racfid=admin1,profiletype=user,cn=racfu01,o=ibm,c=us
-w passwd -b "racfuserid=newuser+racfgroupid=grp222,profiletype=connect,
 cn=racfu01,o=ibm,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=ibm,c=us
racfuserid=NEWUSER
racfgroupid=GRP222
racfconnectauthdate=07/18/05
racfconnectowner=RACFID=ADMIN1,PROFILETYPE=USER,cn=racfu01,o=ibm,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:

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

or

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

For both commands, the results are:

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

racfuserid=NEWUSER+racfgroupid=GRP222,profiletype=CONNECT,cn=racfu01,o=ibm,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):

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

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

Deleting attributes

If a request is made to delete the **racfAttributes** attribute and no values are provided, SDBM generates a command to delete the following values (even if the user does not currently have that value): ADSP, AUDITOR, GRPACC, OIDCARD, OPERATIONS, SPECIAL, UAUDIT. Similarly, a request to delete the **racfConnectAttributes** attribute with no values results in a command to delete the following values: ADSP, AUDITOR, GRPACC, OPERATIONS, SPECIAL. Deleting a specific value for **racfAttributes** or **racfConnectAttributes** requires that the value itself be specified on the delete operation.

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

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

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

```
dn: racfid=user1,profiletype=user,cn=racfu01,o=ibm,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, you would issue an **ldapmodify** command with one of the following files:

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

or

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

Following are some additional examples of deleting attributes:

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

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

Chapter 6. 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*. DIGEST-MD5 is defined in RFC 2831: *Using Digest Authentication as a SASL Mechanism*. Both the CRAM-MD5 and DIGEST-MD5 mechanisms are multi-stage binds where the server sends the client a challenge and then the client sends a challenge response back to the server to complete the authentication. The client challenge response contains a hash of the password entered by the user, the username, and other pieces of data encoded to the specifications of either the CRAM-MD5 or DIGEST-MD5 RFCs.

The CRAM-MD5 and DIGEST-MD5 SASL bind mechanisms are more secure than performing simple binds since the credentials are not passed in clear text. Also, the CRAM-MD5 and DIGEST-MD5 bind mechanisms on the z/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*. 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 backend for CRAM-MD5 and DIGEST-MD5 authentication

The following are considerations for setting up a LDBM 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 entries 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 operation utilities, such as **ldapsearch**, this is done by only specifying the **-D** option.

   b. Only specifying the username in the CRAM-MD5 bind mechanism in your client application. The username that is specified must map to one of the **uid** attribute values in one of the LDBM entries. When using the z/VM LDAP operation utilities, such as **ldapsearch**, this is done by only specifying the **-U** option.
   c. Specifying both the bindDN in the bind request and the username in the CRAM-MD5 bind mechanism in your client application. The username that is specified must map to one of the **uid** attribute values in one of the LDBM entries. The bindDN specified in the bind request must map to the same distinguished name as the username. When using the z/VM LDAP operation utilities, such as **ldapsearch**, this is done by specifying both the **-D** and the **-U** options.

   For more information on the z/VM LDAP operation utilities, see *z/VM: TCP/IP LDAP Administration 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 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 entries. When using the z/VM LDAP operation utilities, such as **ldapsearch**, this is done by only specifying the **-U** option.
   b. Specifying both the username and the authorization DN in the DIGEST-MD5 bind mechanism in your client application. The username that is specified must map to one of the **uid** attribute values in one of the LDBM entries. The authorization DN that is specified must map to the same distinguished name as the username. When using the z/VM LDAP operation utilities, such as **ldapsearch**, this is done by specifying both the **-D** and the **-U** options.

   For more information on the z/VM LDAP operation utilities, see *z/VM: TCP/IP LDAP Administration 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 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” in *z/VM: TCP/IP Planning and Customization*. The z/VM 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.

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: Due to space limitations of the diagram, the entries in the example do not contain all of the necessary information to make them valid directory entries. For example, object classes and required attributes have been left out of many of the entries.

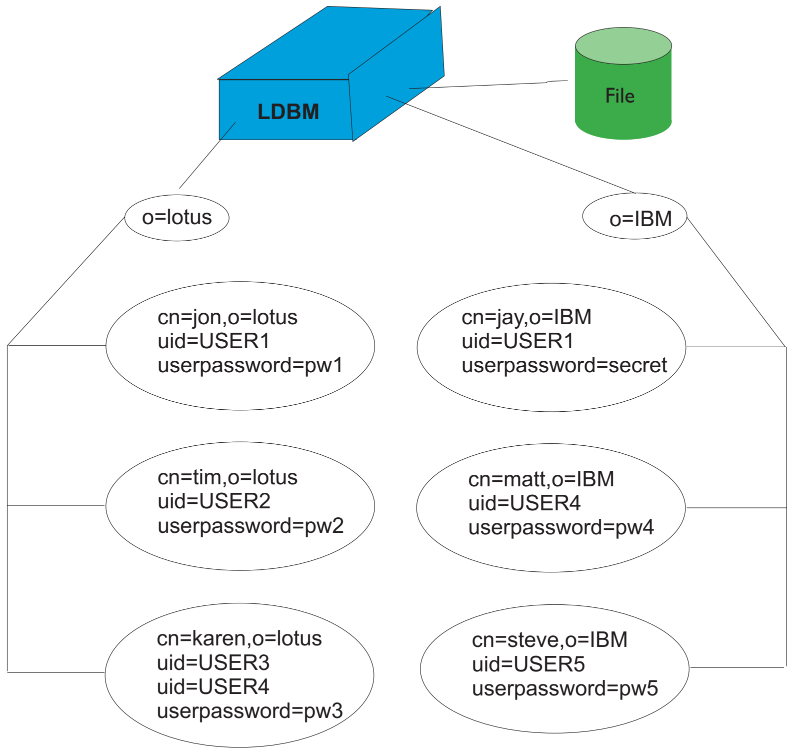


Figure 19. CRAM-MD5 and DIGEST-MD5 authentication example

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 operation utilities, while the bindDN (CRAM-MD5) or authorization DN (DIGEST-MD5) is the **-D** option on the z/VM LDAP operation utilities. See *z/VM:*

TCP/IP LDAP Administration Guide for more details on the LDAP operation utilities. This table assumes that native authentication is not turned on under the subtrees: `o=lotus` and `o=IBM`.

Table 13. Behavior of CRAM-MD5 and DIGEST-MD5 authentication in example

Username	BindDN (CRAM-MD5) or authorization DN (DIGEST-MD5)	Password	Behavior
USER2		pw2	Bind is successful to `cn=tim,o=lotus`
USER2	`cn=tim,o=lotus`	pw2	Bind is successful to `cn=tim,o=lotus`
USER2	`cn=jon,o=lotus`	pw2	Bind is not successful because the bindDN (CRAM-MD5) or authorization DN (DIGEST-MD5) `cn=jon,o=lotus` does not equal the username DN `cn=tim,o=lotus`
USER1		pw1	Bind is not successful, because there are multiple entries with the same username value: `cn=jon,o=lotus` and `cn=jay,o=IBM`
USER1	`cn=jay,o=IBM`	secret	Bind is successful to `cn=jay,o=IBM` because the username DN `cn=jay,o=IBM` equals the bindDN (CRAM-MD5) or authorization DN (DIGEST-MD5) `cn=jay,o=IBM`
USER1	`cn=jon,o=lotus`	pw1	Bind is successful to `cn=jon,o=lotus` because the username DN `cn=jon,o=lotus` equals the bindDN (CRAM-MD5) or authorization DN (DIGEST-MD5) `cn=jon,o=lotus`
USER3		pw3	Bind is successful to `cn=karen,o=lotus`
USER4	`cn=karen,o=lotus`	pw3	Bind is successful to `cn=karen,o=lotus`
USER4	`cn=matt,o=IBM`	pw4	Bind is successful to `cn=matt,o=IBM`
USER3	`cn=karen,o=lotus`	bad	Bind is not successful to username DN `cn=karen,o=lotus` and bindDN (CRAM-MD5) or authorization DN (DIGEST-MD5) `cn=karen,o=lotus` because the password is incorrect.
USER5	`cn=nothere,o=lotus`	pw5	Bind is not successful because the username DN `cn=steve,o=IBM` does not equal the non-existent bindDN (CRAM-MD5) or authorization DN (DIGEST-MD5) `cn=nothere,o=lotus`
BAD		pw1	Bind is not successful because a **uid** value equal to BAD was not found in any of the entries in the LDBM backend.

Chapter 7. 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 8, “Using access control” for additional information on access control checking.

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 31. 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 **ldapsearch** 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* (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=endicott
objectclass: groupOfURLs
cn: ldap_team_dynamic
memberURL: ldap:///o=endicott??sub?(ibm-group=ldapTeam)
```

Dynamic group search filter examples:

A single entry in which the scope defaults to base and the filter defaults to ″objectclass=*″:

```
ldap:///cn=Ricardo,ou=Endicott,o=ibm,c=us
```

The ″In Flight Systems″ team with a scope of one-level and the filter defaults to ″objectclass=*″:

```
ldap:///ou=In Flight Systems,ou=Endicott,o=ibm,c=us??one
```

A subtree search for all the support staff in Endicott:

```
ldap:///ou=Endicott,o=ibm,c=us??sub?title=*Support
```

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

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

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

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

Nested groups

A nested group is defined as a group that references other group entries, which can be static, dynamic, or nested groups. The **ibm-nestedGroup** object class uses the multi-valued attribute called **ibm-memberGroup** to indicate the DNs of the groups that are referenced by the nested group. 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 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=endicott
objectclass: container
objectclass: ibm-nestedGroup
cn: ldap_team_nested
ibm-memberGroup: cn=ldap_team_static,o=endicott
ibm-memberGroup: cn=ldap_team_dynamic,o=endicott
ibm-memberGroup: cn=ldaptest_team_nested,o=endicott
```

Determining group membership

The members of a group entry are determined depending on the type of group. Note that a group can be multiple types (for instance, both dynamic and static).

1. static group: the values of the **member** attribute of the group entry if the object class of the group entry is **accessGroup, accessRole, groupOfNames,** or **ibm-staticGroup**, or the values of the **uniqueMember** attribute if the object class is **groupOfUniqueNames**.
2. dynamic group: the DN of each entry in this LDBM 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 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 only supported on entries within the LDBM 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 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 8, “Using access control.”

ACL restrictions on group gathering

At authentication time, a list is created containing the static, dynamic, and nested groups of which the binding user is a member. No ACL processing is done when reading group entries for group gathering because it is not possible to know what access rights the binding user will have to any of the attributes or subtrees in the directory until all the groups have been fully determined.

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.

```
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
cn: group1
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: person
```

```
cn: nested_grp
sn: group
ibm-memberGroup: cn=dynamic_team,o=Your Company
ibm-memberGroup: cn=group1,o=Your Company
```

Modifying group entries: In order to add a member to a static group, add the user's distinguished name as an additional value for the **member** or **uniqueMember** attribute. Following is an example:

```
ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modStaticGrp.ldif
```

Where modStaticGrp.ldif contains:

```
dn: cn=group1, o=Your Company
changetype: modify
add: member
member: cn=jeff, cn=tim, o=Your Company

dn: cn=group2, o=Your Company
changetype: modify
add: uniqueMember
uniqueMember: cn=joe,o=Your Company
```

In order to remove a member from a static group, remove the user's distinguished name from the set of **member** or **uniqueMember** attribute values in the static group entry. Following is an example:

```
ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modStaticGrp.ldif
```

Where modStaticGrp.ldif contains:

```
dn: cn=group1, o=Your Company
changetype: modify
delete: member
member: cn=jeff, cn=tim, o=Your Company

dn: cn=group2, o=Your Company
changetype: modify
delete: uniqueMember
uniqueMember: cn=joe,o=Your Company
```

In order to add a new search expression to a dynamic group, add the LDAP URL search expression as a value of the **memberURL** attribute. Following is an example:

```
ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modDynamicGrp.ldif
```

Where modDynamicGrp.ldif contains:

```
dn: cn=dynamic_team, o=Your Company
changetype: modify
add: memberURL
memberURL: ldap:///o=Your Company??sub?(employeeType=javaTeam)
```

In order to remove a search expression from a dynamic group entry, the **memberURL** attribute value containing the search expression must be removed from the group entry. Following is an example:

```
ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modDynamicGrp.ldif
```

Where modDynamicGrp.ldif contains:

```
dn: cn=dynamic_team, o=Your Company
changetype: modify
delete: memberURL
memberURL: ldap:///o=Your Company??sub?(employeeType=javaTeam)
```

In order to add a new group reference to an existing nested group entry, add the new group's DN as a value of the **ibm-memberGroup** attribute. Following is an example:

```
ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modNestedGrp.ldif
```

Where modNestedGrp.ldif contains:

```
dn: cn=nested_grp, o=Your Company
changetype: modify
add: ibm-memberGroup
ibm-memberGroup: cn=group2,o=Your Company
```

In order to remove a group reference entry from an existing nested group entry, the **ibm-memberGroup** attribute value containing the group reference DN must be deleted. Following is an example:

```
ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modNestedGrp.ldif
```

Where modNestedGrp.ldif contains:

```
dn: cn=nested_grp, o=Your Company
changetype: modify
delete: ibm-memberGroup
ibm-memberGroup: cn=group2,o=Your Company
```

Deleting group entries: In order to delete a static, dynamic, or nested group entry, delete the directory entry that represents the group. The **ldapdelete** command can be used to perform this delete operation.

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

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

Where deleteGrp.list contains:

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

Examples of querying group membership

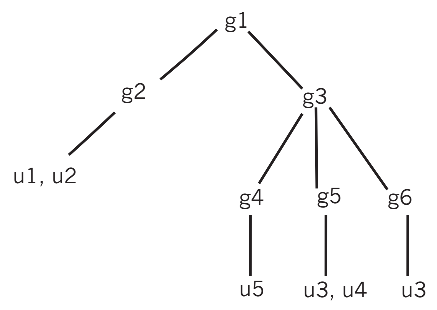


Figure 20. Group hierarchy and membership for the examples

The entries below are used in the following examples:

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

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

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

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

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

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

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

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

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

dn: cn=u3,o=ibm
objectclass: person
aclEntry: access-id:cn=u1,o=ibm:normal:rsc:system:rsc:at.cn:deny:s
aclEntry: access-id:cn=u2,o=ibm:normal:rsc:system:rsc:at.ibm-allGroups:deny:r
aclEntry: group:cn=ANYBODY:normal:rsc:system:rsc
cn: u3
sn: user
userpassword: secret3
```

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

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

dn: cn=u6,o=ibm
objectclass: person
cn: u6
sn: user
userpassword: secret6
```

Note: The **ibm-allMembers** and **ibm-allGroups** attributes are **system** class attributes. The **member** and **cn** attributes are **normal** class attributes.

ibm-allGroups and ibm-allMembers search and comparison examples:

Example 1: This example shows an **ibm-allMembers** attribute search on a static group entry.

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

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

Access checking done for `cn=u6,o=ibm`:

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

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

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

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

Access checking done for `cn=u6,o=ibm`:

1. Read access to the **ibm-allMembers** attribute in `cn=g5,o=ibm`.
2. Search access to the **cn** attribute in the returned entries, `cn=u3,o=ibm` and `cn=u4,o=ibm`, from the search filter specified in the **memberURL** attribute.

Note: memberURL attribute access rights do not matter.

Example 3: This example shows an **ibm-allMembers** attribute search on a nested group entry.

```
ldapsearch –L –D "cn=u6,o=ibm" –w secret6 –b "cn=g3,o=ibm" "objectclass=*" ibm-allMembers

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

Access checking done for `cn=u6,o=ibm`:

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

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

Note: **ibm-memberGroup** access rights do not matter.

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

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

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

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

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

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

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

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

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

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

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

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

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

Access checking done for `cn=u1,o=ibm`:

1. Read access to the **ibm-allMembers** attribute in `cn=g5,o=ibm`.
2. Search access to the **cn** attribute in the returned entries, `cn=u3,o=ibm` and `cn=u4,o=ibm`, from the search filter specified in the **memberURL** attribute. However, search access has been denied on the **cn** attribute of `cn=u3,o=ibm` therefore it is not added as an **ibm-allMembers** attribute value.

Example 7: This example shows an **ibm-allMembers** comparison operation on a dynamic group entry.

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

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

1. Compare access to the **ibm-allMembers** attribute in `cn=g5,o=ibm`.
2. Search access to the **cn** attribute on the returned entries, `cn=u3,o=ibm` and `cn=u4,o=ibm`, from the search filter specified in the **memberURL** attribute.

Example 8: This example shows an **ibm-allGroups** attribute search where the user belongs to dynamic and nested group entries.

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

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

Access checking done for `cn=u6,o=ibm`:

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

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

Example 9: This example shows an **ibm-allGroups** attribute search where the user belongs to static and nested group entries.

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

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

Access checking done for `cn=u1,o=ibm`:

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

Since `cn=g1,o=ibm` has an **ibm-memberGroup** attribute value of `cn=g2,o=ibm`, `cn=g1,o=ibm` is added as an **ibm-allGroups** attribute value.

Example 10: This example shows an **ibm-allGroups** attribute search where the user being searched belongs to static and nested group entries. The bound user has read authority to the **ibm-allGroups** attribute of the user being searched, but does not have read authority on the **member** attribute in the static group entry.

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

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

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

1. Read access to the **ibm-allGroups** attribute in `cn=u2,o=ibm`.
2. Read access to the **member** attribute of `cn=g2,o=ibm` is denied. Therefore, `cn=g2,o=ibm` is not added as an **ibm-allGroups** attribute value.

Example 11: This example shows an **ibm-allGroups** search where the bound user does not have read authority on the **ibm-allGroups** attribute.

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

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

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

1. Read access to the **ibm-allGroups** attribute in `cn=u4,o=ibm` is denied. Therefore, no **ibm-allGroups** attribute values are added.

Example 12: This example shows an **ibm-allGroups** comparison operation where the bound user is allowed to determine that a user belongs to a nested group entry.

```
ldapcompare -D "cn=u2,o=ibm" -w secret2 "cn=u3,o=ibm" "ibm-allGroups=cn=g1,o=ibm"
ldap_compare: Compare true
```

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

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

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

Chapter 8. Using access control

Access control of information in the LDAP server is specified by setting up Access Control Lists (ACLs). LDBM or GDBM ACLs provide a means to protect information stored in an LDAP directory. Administrators use ACLs to restrict access to different portions of the directory, or specific directory entries. When using the LDBM or GDBM 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 or GDBM backend using RACF-style user and group DNs. This controls access to LDBM or GDBM 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 14 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 14. ACL and entry owner attributes

ACL attributes	
aclEntry	This is a multi-valued attribute that contains the names and permissions associated with those names that have access to information in the directory entry (or the entry along with the subtree of information below the entry, depending on the setting of the **aclPropagate** attribute).
aclPropagate	This is a single-valued boolean attribute which indicates whether the **aclEntry** information applies only to the directory entry it is associated with or to the entire subtree of information including and below the directory entry it is associated with. Note that propagation does not apply to entries that have an explicit **aclEntry** defined for the entry and that propagation stops at the next propagating ACL (`aclPropagate=TRUE`) that is encountered in the directory subtree.

Table 14. ACL and entry owner attributes (continued)

aclSource	This is a single-valued attribute that is managed by the LDAP server and cannot be changed by the **ldapmodify** command. This attribute, accessible for any directory entry, indicates the distinguished name of the entry that holds the ACL that applies to the entry. This attribute is useful in determining which propagating ACL is used to control access to information in the directory entry.
Entry owner attributes	
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 **ldapmodify** command. This attribute indicates the distinguished name of the entry that holds the **entryOwner** that applies to the entry. This attribute is useful in determining which propagating **entryOwner** is used to control access to information in the directory entry.

aclEntry attribute

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:

```
[access-id:|group:|role:]subject_DN[granted_rights]
```

The *subject_DN* is any valid DN which represents the object (entry) to which privileges are being granted. The DN ends when the first granted rights keyword is detected.

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

```
[:object:[grant:|deny:]object_rights_list] [:normal:[grant:|deny:]attr_rights_list]
[:sensitive:[grant:|deny:]attr_rights_list] [:critical:[grant:|deny:]attr_rights_list]
[:restricted:[grant:|deny:]attr_rights_list] [:system:[grant:|deny:]attr_rights_list]
[:at.attr_name:[grant:|deny:]attr_rights_list]
```

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 being granted access.

group The distinguished name of the group entry being granted access.

role The distinguished name of the group entry being granted access.

Access control groups can be either static, dynamic, or nested groups. The following object classes are evaluated as group entries for LDBM: **ibm-staticGroup, groupOfNames, groupOfUniqueNames, accessRole, accessGroup, ibm-dynamicGroup, groupOfUrls,** and **ibm-nestedGroup**. See Chapter 7, “Static, dynamic, and nested groups” for additional information on static, dynamic, and nested groups.

Privilege attributes take the form of *type*:*name* where *type* refers to either **access-id**, **group**, or **role** and *name* is the distinguished name.

Note: The distinguished name that is used need not be the name of an entry in the directory. The distinguished name is the name that represents the user that has authenticated to the directory server.

The *type*: portion of this clause is optional.

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.

Examples

In this example, the DN type is `access-id` and the DN itself is `cn=personA, ou=deptXYZ, o=IBM, c=US`.

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

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

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

This is an example of how to use a RACF identity 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 15. Permissions which apply to an entire entry

Add	Add an entry below this entry
Delete	Delete this entry

Table 16. Permissions which apply to attribute access classes

Read	Read attribute values
Write	Write attribute values
Search	Search filter can contain attribute type
Compare	Compare attribute values

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
```

See Access determination for information on how the **aclEntry** values are used to determine access.

The **aclEntry** attribute values are defined as directory strings but contain a distinguished name as one of the components of the attribute value. When performing a search using one of these attributes, only the distinguished name is used in the search filter and the value is normalized following the matching rules for a distinguished name. Two **aclEntry** attributes are considered to be the same if they have the same distinguished name.

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 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.

entryOwner attribute

Each entry has an associated **entryOwner**. The **entryOwner** might be a user or a group, similar to what is allowed within the **aclEntry**. However, the **entryOwner** subject has certain privileges over the entry.

Entry owners are, in essence, the administrators for a particular entry. They have full access on that particular entry, similar to the administrator DN. Note that the administrator DN has full permission on every entry in the database. Each **entryOwner** attribute value is a distinguished name. However, for compatibility with previous releases, the distinguished name can be preceded with **access-id:**, **group:**, or **role:**.

Note: The distinguished name that is used need not be the name of an entry in the directory. The distinguished name is the name that represents the user that has authenticated to the directory server.

Entry owners are not constrained by permissions given in the **aclEntry**. They have complete access to any entry attribute, and can add and delete entries as desired.

Entry owners, the administrator DN, and users who have write permission for **restricted** attributes are the only people who are allowed to change the attributes related to access control. If a backend is defined as a peer or read-only replica, only the administrator DN and the peerserver DN or masterserver DN can set the access control attributes within the backend directory.

The **entryOwner** attribute values are defined as directory strings but contain a distinguished name as one of the components of the attribute value. When performing a search using one of these attributes, only the distinguished name is used in the search filter and the value is normalized following the matching rules for a distinguished name. Two **entryOwner** attributes are considered to be the same if they have the same distinguished name.

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.

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 administrator DN, 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 administrator DN is assigned to the entry. The default value cannot be deleted. If an **entryOwner** value is later added to the entry, explicitly or by inheritance, the entire default **entryOwner** value is replaced. The LDAP server sets the value of the **ownerSource** attribute to 'default' when the entry is using the default owner.

Initializing ACLs with GDBM

When the LDAP sever is started with GDBM configured for the first time, the LDAP server creates the change log suffix entry, cn=changelog. The suffix entry is created with an **aclEntry** and **entryOwner** value that allows access only to the LDAP administrator and propagates the **aclEntry** and **entryOwner** values. Only the **aclEntry** and **entryOwner** attributes can be modified. Change log entries cannot be modified to override the inherited ACL values from the change log suffix entry.

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, which allows read access to anybody. Therefore, only the LDAP administrator can update the schema. The **aclEntry** and **entryOwner** values can be modified.

Access determination

The same distinguished name (DN) may be granted different access permissions to an entry, from specific access permissions to the DN and from group memberships (including the authenticated and anybody groups). The LDAP server uses the following algorithm to determine which permissions to grant a DN based on the values in the **aclEntry** attribute:

- if there is a specific value for the DN, the DN gets those permissions only

- else if there is a cn=this value and the DN is the distinguished name of the entry, the DN gets those permissions only
- else if there are one or more group values that the DN is a member of, the DN gets the union of the permissions for those groups
- else if there is a cn=authenticated value and the DN is authenticated to the directory with an LDAP bind operation, the DN gets those permissions only
- else if there is a cn=anybody value, the DN gets those permissions only
- otherwise the DN gets no permissions

Each of the access permissions is discrete. One permission does not imply another.

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 precedence over a grant permission (for the same item specified) while attribute-level permissions take precedence over access-class permissions.

Following are examples for permissions:

Example 1

```
aclEntry: group:cn=Anybody:normal:rsc
```

In this example, unauthenticated (anonymous) users have permission to read, search and compare all attributes within the **normal** attribute access class. ACL entry values for unauthenticated users use pseudoDN `cn=Anybody`.

Example 2

```
aclEntry: access-id:cn=personA,ou=deptXYZ,o=IBM,c=US:object:ad:normal:rwsc:sensitive:rwsc:critical:rsc
```

In this example, the user corresponding to `cn=personA, ou=deptXYZ, o=IBM, c=US` has permission to add entries below the entry, to delete the entry, to read, write, search and compare both **normal** and **sensitive** attributes, and to read, search and compare **critical** attributes.

Example 3

```
aclEntry: group:cn=Authenticated:normal:rwsc:sensitive:rwsc
```

In this example, users who have authenticated to the directory where a specific **aclEntry** value does not apply, will be allowed to read, write, search, and compare, **normal** and **sensitive** attributes in the directory entry.

Example 4

```
aclEntry: cn=Tim,dc=yourcompany,dc=com:at.cn:deny:w:normal:rwsc
```

In this example, `cn=Tim,dc=yourcompany,dc=com` is granted read, write, search, and compare to **normal** attributes except for the **cn** attribute in which write access is denied. Note that the following ACL entry results in the same access:

```
aclEntry: cn=Tim,dc=yourcompany,dc=com:normal:rwsc:at.cn:deny:w
```

The evaluation of the permissions clauses is based on precedence, not order in the ACL entry value(s).

Example 5

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

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

Example 6

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

In this example, a member of group1 only would be granted read, write, search, and compare to **normal** attributes. A member of both group1 and group2 would be granted read, write, search, and compare to **normal** and **sensitive** attributes, excluding write access to the **cn** attribute. This is an example where a member of both groups is granted access to less information than what is granted to each of the two groups individually.

Example 7

```
aclEntry: access-id:cn=Tim,dc=yourcompany,dc=com:normal:rwsc:at.cn:rsc
```

In this example, `cn=Tim,dc=yourcompany,dc=com` is granted read, write, search, and compare on **normal** attributes and read, search, and compare on the **cn** attribute. Note that `cn=Tim,dc=yourcompany,dc=com` will also have write access to the **cn** attribute by virtue of **cn** having an access class of **normal**.

Example 8

```
aclEntry: access-id:cn=Tim,dc=yourcompany,dc=com:normal:rwsc:at.cn:deny:rsc
```

In this example, `cn=Tim,dc=yourcompany,dc=com` is granted read, write, search, and compare on **normal** attributes and denied read, search, and compare on the **cn** attribute. Note that `cn=Tim,dc=yourcompany,dc=com` will still have write access to the **cn** attribute by virtue of **cn** having an access class of **normal**.

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 to which 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 to which 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 to which 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, let the **aclEntry** be

```
group:cn=Anybody:normal:rsc:sensitive:c:critical:c
```

and let a client perform an anonymous search

```
ldapsearch -b "c=US" "cn=LastName" title userpassword telephoneNumber
```

where `title` is a **normal** attribute, `telephoneNumber` is a **sensitive** attribute, and `userpassword` is a **critical** attribute. Users performing anonymous searches are given the permission granted to the **cn=Anybody** group. In this example, permission exists to the filter since **cn** is in the **normal** attribute access class, and **cn=Anybody** has **s** (search) permission to the **normal** attribute access class. What is returned however, is only the **title** attribute for any matching entry. The **telephoneNumber** and **userPassword** attributes are not returned since **cn=Anybody** does not have read permissions on the **sensitive** and **critical** attribute access classes.

Propagating ACLs

ACLs can be set on any entry in the hierarchy. ACLs can propagate down through the directory hierarchy. These ACLs, called propagating ACLs, have the **aclPropagate** attribute set to **TRUE**. All descendents of this entry will inherit the ACL set at that point, unless overridden. In order to specify an ACL different from that of its parent, a new ACL must be explicitly set.

When setting the new ACL, there is again a choice of whether to propagate the ACL. If set to **TRUE**, the ACL will propagate down to all descendants. If set to **FALSE**, the ACL is not propagated; it instead becomes an override ACL. The ACL is not propagated down through the hierarchy, but instead applies only to the one particular entry that it is associated with within the hierarchy. If unspecified, **aclPropagate** is set to **TRUE**.

An entry without an explicit ACL receives its ACL from the nearest propagating ancestor ACL. 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 based on the **ownerPropagate** attribute.

Example of propagation

Following is the explicit ACL for entry `ou=deptXYZ, o=IBM, c=US` :

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

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

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

In this example, a propagating ACL has been set on `ou=deptXYZ, o=IBM, c=US`. No ACL has been set on the descendant `cn=personA, ou=deptXYZ, o=IBM, c=US`. Therefore, the descendant inherits its ACL value from the nearest ancestor with a propagating ACL. This happens to be `ou=deptXYZ, o=IBM, c=US`, which is reflected in the **aclSource** attribute value. The **aclEntry** and **aclPropagate** values are identical to those values in the explicit propagating ACL set at `ou=deptXYZ, o=IBM, c=US`.

Examples of overrides

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

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

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

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

Note that in the explicit ACLs above, **aclSource** is the same as the entry DN. This attribute is generated and managed by the LDAP server; it cannot be set when modifying ACLs.

Following is an implicit ACL for entry `cn=personA, ou=deptXYZ, o=IBM, c=US`:

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

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

The descendant `cn=personA, ou=deptXYZ, o=IBM, c=US` inherits its ACL value from the nearest ancestor with a propagating ACL (which is `o=IBM, c=US` as reflected in the **aclSource**). The ACL on `ou=deptXYZ, o=IBM, c=US` is not used because **aclPropagate** is **FALSE**.

Other examples

In these examples, the administrator DN will be `cn=admin, c=US`.

The following example shows the default ACL:

```
aclPropagate:  TRUE
aclEntry: group:cn=Anybody:normal:rsc:system:rsc
aclSource: default
ownerPropagate:  TRUE
entryOwner:  access-id:cn=admin,c=US
ownerSource:  default
```

The following example shows a typical ACL for entry `cn=personA, ou=deptXYZ, o=IBM, c=US`:

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

This is an inherited ACL and an inherited owner. Both owner properties and ACL properties are inherited from entry `ou=deptXYZ, o=IBM, c=US`. In this example, members of group `cn=deptXYZRegs, o=IBM, c=US` have permission to read, search and compare 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 on attributes in the **critical** attribute access class. Unauthenticated, as well as all other bound users, have permission to read, search, and compare attributes in the **normal** and **system** attribute access classes only. The `personA` has add and delete permission on the entry; read, write, search, and compare permissions on **normal** and **sensitive** attributes; and read, search, and compare permission on **critical** attributes. The `deptXYZMgr` had full access to the entry since it is the owner of the entry. As always, the administrator also has unrestricted access to the entry.

Access control groups

Access control groups provide a mechanism for applying the same **aclEntry** 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 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 7, “Static, dynamic, and nested groups” for more information on static, dynamic, and nested groups.

Associating DNs and access groups 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 backend, an SDBM backend, or in a client operation plug-in extension.
- 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 backend, or in a client operation plug-in extension. See Chapter 6, “CRAM-MD5 and DIGEST-MD5 authentication,” on page 81 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 backend, or in a client operation plug-in extension. See Chapter 6, “CRAM-MD5 and DIGEST-MD5 authentication,” on page 81 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 backend or in a client operation plug-in extension corresponding to this DN.

After the bind DN is determined, the DNs of the groups that the bound user belongs to are also added to the bind information. 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 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.

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 **ldapsearch** command, as shown in the following example:

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

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

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

ACL information (**aclEntry**, **aclPropagate**, **aclSource**, **entryOwner**, **ownerPropagate**, and **ownerSource**) is returned for all entries. For those entries that contain ACLs, the **aclSource** and **ownerSource** attributes contain the same DN as the entry DN. For those entries that do not contain ACLs, the **aclSource** and **ownerSource** attributes contain distinguished names of the entries that contain the ACL information (**aclEntry** and **entryOwner**) that are used for access control checking of information in that entry.

Notes:

1. It is possible for the **aclSource** and **ownerSource** attributes to contain the value `default`. This is not a distinguished name but rather represents that the ACL that applies to the entry is the default ACL.
2. If the tree is larger than the **sizeLimit** option in the LDAP server configuration file or on the search command, then not all entries are returned. See the **sizeLimit** configuration option in "Configuring the LDAP Server" in *z/VM: TCP/IP Planning and Customization* for more information.

You can also use the same method to get the ACL information for a portion of the namespace by specifying the **-b** *searchbase* parameter on the **search** command, where *searchbase* is the starting point for the search.

Creating and managing access controls

To create and update ACLs in LDBM, GDBM, or the schema entry, use a tool implementing **ldap_modify** APIs, such as **ldapmodify**. The **ldapmodify** utility allows creation, modification, and deletion of any set of attributes that are associated with an entry in the directory. Since access control information is maintained as a set of additional attributes within an entry, **ldapmodify** is a natural choice for administering access control information in LDBM, GDBM, or the schema entry.

See *z/VM: TCP/IP User's Guide* for details on using the utilities, such as **ldapmodify**.

Creating an ACL

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

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

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

Figure 21 shows how to add a propagating ACL with three **aclEntry** values to an existing entry replacing any current **aclEntry** value.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f newAcl.ldif
```

Where `newAcl.ldif` contains:

```
dn: cn=tim, o=Your Company
changetype: modify
replace: aclEntry
aclEntry: cn=jeanne, o=Your Company:
 normal:rsc:sensitive:rsc:critical:rsc
aclEntry: cn=jeff, cn=tim, o=Your Company:normal:rsc
aclEntry: cn=tim, o=Your Company:
 normal:rwsc:sensitive:rwsc:critical:rwsc
-
```

Figure 21. Example of adding propagating ACL to existing entry in directory

The ACL added in Figure 21 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 **ldapmodify** commands are performed from the same system on which the LDAP server is running and that the LDAP server is listening on TCP/IP port 389. Refer to the **ldapmodify** command description in *z/VM: TCP/IP User's Guide* for more details on the **-h**, **-p**, **-D**, and **-w** command-line options. The ACL attributes can be updated from any LDAP client as long as the user performing the updates has the proper authorization to update the ACL information.

The ACL attributes are defined to be in a special access class called **restricted**. 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 22 shows an example of adding an ACL so that `cn=jeanne, o=Your Company` can update the ACL information:

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f newAcl.ldif
```

Where `newAcl.ldif` contains:

```
dn: cn=jeanne, o=Your Company
changetype: modify
replace: aclEntry
aclEntry: cn=jeanne, o=Your Company:
 normal:rsc:sensitive:rsc:critical:rsc:restricted:rwsc
aclEntry: cn=jeff, cn=tim, o=Your Company:normal:rsc
aclEntry: cn=tim, o=Your Company:
 normal:rsc
-
add: aclPropagate
aclPropagate: TRUE
-
```

Figure 22. Example of adding propagating ACL to existing entry in the directory.

The ACL added in Figure 22 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 23 shows an example of adding a non-propagating ACL. A non-propagating ACL applies only to the entry to which it is attached and not to the subtree of information that might be stored below the entry in the directory.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f newAcl.ldif
```

Where `newAcl.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
replace: aclEntry
aclEntry: cn=tim, o=Your Company:normal:rwsc:sensitive:rwsc:
 critical:rwsc:restricted:rwsc
aclEntry: cn=jeff, cn=tim, o=Your Company:normal:rwsc:
 sensitive:rwsc:critical:rwsc
aclEntry: cn=jeanne, o=Your Company:normal:rsc
-
replace: aclPropagate
aclPropagate: FALSE
-
```

Figure 23. Example of setting up a non-propagating ACL

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

Modifying an ACL

Once an ACL exists for an entry in the directory, it may have to be updated. To do this, the **ldapmodify** command is used. As described earlier in This topic, while the **ldapmodify** command is used in these examples, what is really being used is an LDAP client application, issuing LDAP modify operations to the LDAP server. 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 24, Figure 25, and Figure 26 show examples of these modifications, respectively.

Access determination shows how an additional **aclEntry** value is added to existing ACL information.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modAcl.ldif
```

Where `modAcl.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
add: aclEntry
aclEntry: cn=dylan, cn=tim, o=Your Company:
 normal:rwsc:sensitive:rwsc:critical:rwsc:restricted:rwsc
-
```

Figure 24. Example of adding an aclEntry attribute value

In Figure 24, `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 25 shows how to modify an existing ACL to be non-propagating instead of propagating.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modAcl.ldif
```

Where `modAcl.ldif` contains:

```
dn: cn=tim, o=Your Company
changetype: modify
replace: aclPropagate
aclPropagate: FALSE
-
```

Figure 25. Example of modifying aclPropagate attribute

In Figure 25, 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 26 shows how to remove an **aclEntry** value from existing ACL information:

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modAcl.ldif
```

Where `modAcl.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
delete: aclEntry
aclEntry: cn=dylan, cn=tim, o=Your Company
-
```

Figure 26. Example of removing a single aclEntry attribute value

In Figure 26, 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 **ldapmodify** command to delete the entire attribute (all values) from the entry.

Figure 27 shows an example of deleting an ACL from an entry.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f delAcl.ldif
```

Where `delAcl.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
delete: aclEntry
-
delete: aclPropagate
-
```

Figure 27. Example of deleting an ACL from an entry

In Figure 27, 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 **ldapmodify** command is very powerful, all the possible ways of adding the **entryOwner** and **ownerPropagate** information cannot be shown here. The examples shown here describe the more common uses of the **ldapmodify** command to add entry owner information.

Figure 28 shows how to add a propagating entry owner with two **entryOwner** values to an existing entry.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f newOwn.ldif
```

Where `newOwn.ldif` contains:

```
dn: cn=tim, o=Your Company
changetype: modify
replace: entryOwner
entryOwner: cn=joe, o=Your Company
entryOwner: cn=carol, o=Your Company
-
```

Figure 28. Example of adding a propagating set of entry owners to existing entry in the directory

The entry owners added in Figure 28 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 **ldapmodify** commands are performed from the same system on which the LDAP server is running and that the LDAP server is listening on TCP/IP port 389. Refer to the **ldapmodify** command description in *z/VM: TCP/IP User's Guide* for more details on the **-h**, **-p**, **-D**, and **-w** command-line options. The entry owner attributes can be updated from any LDAP client as long as the user performing the update has the proper authorization to update the entry owner information.

The entry owner attributes, like the ACL attributes, are defined to be in a special access class called **restricted**. 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 22 for an example of allowing users other than the LDAP administrator the ability to update entry owner information.

Figure 29 shows an example of adding a non-propagating entry owner. A non-propagating entry owner applies only to the entry to which it is attached and not to the subtree of information that might be stored below the entry in the directory.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f newOwn.ldif
```

Where `newOwn.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
replace: entryOwner
entryOwner: cn=george, o=Your Company
entryOwner: cn=jane, o=Your Company
-
replace: ownerPropagate
ownerPropagate: FALSE
-
```

Figure 29. Example of setting up a non-propagating entry owner

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

Modifying an owner for an entry

Once an entry owner exists for an entry in the directory, it may have to be updated. To do this, the **ldapmodify** command is used. As described earlier in This topic, while the **ldapmodify** command is used in these examples, what is really being used is an LDAP client application, issuing LDAP modify operations to the LDAP server. 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 30, Figure 31, and Figure 32 show examples of these modifications, respectively.

Figure 30 shows how an additional **entryOwner** value is added to existing entry owner information.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modOwn.ldif
```

Where `modOwn.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
add: entryOwner
entryOwner: cn=george, o=Your Company
-
```

Figure 30. Example of adding an entryOwner attribute value

In Figure 30, `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 31 shows how to modify an existing entry owner to be non-propagating instead of propagating.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modOwn.ldif
```

Where `modOwn.ldif` contains:

```
dn: cn=tim, o=Your Company
changetype: modify
replace: ownerPropagate
ownerPropagate: FALSE
-
```

Figure 31. Example of modifying the ownerPropagate attribute

In Figure 31, 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 32 shows how to remove an **entryOwner** value from existing entry owner information:

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modOwn.ldif
```

Where `modOwn.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
delete: entryOwner
entryOwner: cn=george, cn=tim, o=Your Company
-
```

Figure 32. Example of removing a single entryOwner Attribute value

In Figure 32, 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 **ldapmodify** command to delete the entire attribute (all values) from the entry.

Figure 33 shows an example of deleting an entry owner set from an entry.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f delOwn.ldif
```

Where `delOwn.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
delete: entryOwner
-
delete: ownerPropagate
-
```

Figure 33. Example of deleting an entry owner set from an entry

In Figure 33, 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 7, "Static, dynamic, and nested groups" 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 34 shows how a group can be added to the **aclEntry** information in an existing access control specification for an entry. Figure 35 shows how a group can be added as an **entryOwner** to an existing entry owner specification for an entry.

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modAcl.ldif
```

Where `modAcl.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
add: aclEntry
aclEntry: group:cn=group1, o=Your Company:normal:rwsc:sensitive:rsc
-
```

Figure 34. Example of adding a group to access control information

```
$ ldapmodify -h 127.0.0.1 -D "cn=admin" -w xxxx -f modOwn.ldif
```

Where `modOwn.ldif` contains:

```
dn: cn=jeff, cn=tim, o=Your Company
changetype: modify
add: entryOwner
entryOwner: cn=group1, o=Your Company
-
```

Figure 35. Example of adding a group to entry owner information

Chapter 9. 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 replication:

- In peer to peer replication, each LDAP peer server is a read-write server. Updates processed on one peer server are replicated to all the other peer servers. Peer servers are read-write to all users.

  Note: The z/VM support for peer to peer replication is provided for failover support purposes. There is no support for resolving conflicting simultaneous updates on multiple peer servers, which can cause a failure of replication. As a result, updates should be targeted to one peer server at a time.
- In read-only replication, a single read-write LDAP server (the master) replicates the updates it processes to a set of read-only replica servers.

  Master
  : All changes to the 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.

A replication network can contain both peer replica servers and read-only replica servers. In this case, each peer server must act as a master to each read-only replica (in addition to being a peer to all the peer servers), so that updates that occur on any peer server are replicated to all the other peer and read-only replicas in the network.

Replication is supported when the servers involved are running in single-server. Refer to “Determining Operational Mode” for more information about server operating modes.

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

ibm-entryuuid replication

Replication of the **ibm-entryuuid** will be 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

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 replication

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

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.

Replicating server

In order for the replication process to occur, the following must happen:
- The replicating server (master or peer) must be aware of each replica that is to receive the change information.
- Each read-only and peer server must be aware of the replicating servers for the directory that it serves. See LDAP update operations on read-only replicas 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 find 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 Replication error log for more information about the set aside count.

Table 17. 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.ibm.com`

replicaBindDN	Specifies the LDAP distinguished name that the replicating server uses to bind to the replica when sending directory updates. The **replicaBindDN** and the **masterServerDN** or **peerServerDN** in the replica's LDAP server configuration file must have the same value. Example: `replicaBindDN: cn=Master`
replicaCredentials	Contains the authentication information needed for the replicating server to authenticate to the replica using the **replicaBindDN**. The **replicaCredentials** attribute value will be encrypted if the **secretEncryption** option is specified in the LDAP server configuration file. This improves directory security since 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`
cn	Forms the RDN of the LDAP distinguished name of the **replicaObject** entry. Example: `cn: myReplica`

In the examples in Table 17, when the replicating server receives and successfully finishes an update request, the update is also sent to `myMachine.ibm.com` on port 389 (the default port). The replicating server performs a bind operation using the DN of `cn=Master` and password of `secret`. See “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 could better describe the replica server, and it could listen on a different port then the default port of 389. Examples of adding a description and changing the port to 400 are shown in Table 18, which describes the optional attributes of **replicaObject**.

Table 18. 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 may contain information related to this entry. Example: `seeAlso: cn=Alternate Code, ou=Software, o=IBM, c=US`
replicaBindMethod	Identifies the bind method to be used. If it is specified, it must be set to **simple**. Example: `replicaBindMethod: simple`

Replication only supports simple authentication. SASL EXTERNAL, GSSAPI, DIGEST-MD5, and CRAM-MD5 bind mechanisms are not supported as valid replication bind mechanisms.

There are several additional attributes that affect error handling during replication. See Replication error log for more information on 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 19 describes these attributes.

Table 19. Additional optional replication attributes

Attribute	Description and example
ibm-slapdLog	Specifies the file name of the replication error log. This must be an OpenExtensions 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 replication errors that will be set aside in the 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 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 prior to 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 will allow 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.ibm.com
replicaBindDn: cn=Master
replicaCredentials: secret
replicaPort: 400
```

```
replicaUseSSL: FALSE
description: Replica machine in the fourth floor lab
ibm-slapdLog: ro1.errlog
ibm-slapdReplMaxErrors: 5
```

Searching a replica entry

Most of the attributes in a replica entry are operational attributes. When searching a replica entry, the operational attributes are not included in the output unless they are specified in the attributes to be returned. The following command searches for all replica entries in a suffix and returns the complete replica entries in LDIF format:

```
ldapsearch -h ldaphost -p ldapport -D bindDN -w bindPW -L -b "suffix"
 "objectclass=replicaObject" "*" replicaHost replicaBindDN replicaCredentials
 replicaPort replicaUpdateTimeInterval replicaUseSSL replicaBindMethod
```

Displaying 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.

Maintenance mode

Maintenance mode is the LDAP server setup mode for replication. This mode restricts access to the backends in an LDAP server to allow replica backends to be primed for replication. Access to the backends is as follows:

- read-only replica backend: The **masterServerDN** for the replica and the **adminDN** have unrestricted access
- peer replica backend: The **peerServerDN** for the replica and the **adminDN** have unrestricted access
- non-replica backends (including the schema entry): The **adminDN** has 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. No ACL checking is done for any operations from **adminDN**. In addition, the **adminDN** has the capability in maintenance mode to modify attributes that are read-only and are normally only set by the LDAP server, such as **ibm-entryuuid**.

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. When in maintenance mode, they cannot update the LDAP server schema at all. **adminDN** can always update the schema.

Pending replication entries are replicated to the other replica servers, but updates performed when in maintenance mode are not replicated.

Specify the **-m** option on the server startup command to start the LDAP server in maintenance mode.

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.

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 the LDAP Version 3 protocol.

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 the **ds2ldif** utility (see "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 using the administrator DN to run **ldapmodify**.
4. Using the administrator DN, run **ldapadd** 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.
6. Transport the LDIF file created in step 2 to the replica server's location.
7. Load the LDIF file from 6 into the replica server. This can be done using the administrator DN to run **ldapadd** 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 the administrator DN to run **ldapdelete** 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

Notes:

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 on the **listen** option, see “Step 7. 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 since 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 replication for information about securing a directory.

Changing a read-only replica to a master

When using read-only replication, it may become desirable to change one of the read-only replicas to be the master. Perhaps the machine where the replica server is installed is being upgraded, and 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 out-of-sync conditions.
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 the administrator DN, unload all the replica entries (entries that describe replica servers) from the master server. Use a search command similar to the one shown in Searching a replica entry 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 the administrator DN, run **ldapdelete** to remove the replica entries from the master.
5. Using the 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 7. Create and Customize the LDAP Configuration File (DS CONF)” in *z/VM: TCP/IP Planning and Customization* for more information on 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 on alternatives to specifying the **masterServerPW** option.

10. Start the new master server and new replica server (if the original master became a replica server).

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: Peer to peer replication uses the same replica entry attribute values as shown in Replica server. The instructions in Adding replica entries in LDBM also apply to peer replicas.

A peer to peer replication environment can be as simple as two LDAP servers that are peers to each other, or as complicated as several LDAP servers, where some servers are read-only replication servers and the other servers are peer replication servers. Every peer replication server must replicate to all other peer and read-only replication servers.

Server configuration

The **peerServerDN** and **peerServerPW** options in the backend section of the LDAP server configuration file are used to configure peer to peer replication. See "Step 7. Create and Customize the LDAP Configuration File (DS CONF)" in *z/VM: TCP/IP Planning and Customization* for more information.

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.

Conflict resolution

Minimal conflict resolution is done in a peer environment. For example, if peer replication server A receives an update to entry E at the same moment that peer B receives a delete of the same entry, replication can stall on server A. Ensure that your peer servers are not receiving conflicting operations. To avoid replication stalling, set up a replication error log to set aside replication errors. See Replication error log 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 may be necessary for you to add a peer replica for a backend to an existing server or set of servers. These servers can be standalone 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**). Refer to Populating a replica 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 would 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 may 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 "Establishing 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 may 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 replication

SSL/TLS can be used to communicate between a replicating server (master or peer) and a replica server (read-only or peer).

Replica server with SSL/TLS enablement

Set the replica server up for SSL/TLS just like a normal SSL/TLS server. It needs its own public-private key pair and certificate, and the 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 like an SSL/TLS client to the replica server.

To set up the replicating server, you must:

1. Run the **gskkyman** utility, this time as if you were a client (see “SSL Certificate Management” in *z/VM: TCP/IP User’s Guide*). You should use the same key directory file that contains the replicating server’s key pair and certificate. Receive the replica’s self-signed certificate and mark it as trusted.
2. In the LDAP server configuration file on the replicating server:
   - Set **sslKeyRingFile** to the replica key directory file name created above.
   - Set **sslKeyRingFilePW** to the password for the replica key directory file, or set **sslKeyRingPWStashFile** to the file name where the password is stashed.
3. 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.

Since the replicating server acts like an SSL/TLS client to the replica server, the replicating server binds with the replica server. The bind method used is **simple** bind. The SASL external bind method is not supported for replication.

Replication error log

A replication error log holds information on each error that occurs during replication. To avoid stalling 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, 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 replication stalls for that replica.

The replication error log is used to correct replication in two ways:
- Use the error information to determine why replication failed.
- Invoke **ldapmodify** to run the error log on the replica server, after resolving the 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:

```
#(070102 03:35:46.910816): modify operation failed for cn=IBMUSER01,
  O=YOUR COMPANY to 9.57.1.198:3389, rc=32
# R004071 DN 'cn=IBMUSER01,O=YOUR COMPANY' does not exist (ldbm_process_request)
# setting change aside.

dn: cn=IBMUSER01, O=YOUR COMPANY
changetype: modify
replace: sn
sn: Fred Smith
```

The error log consists of three messages, each using one or more lines:
1. Message one indicates when the error occurred, the entry, and replica server.
2. Message two is the error message returned by the replica server.
3. Message three indicates what is being done. If the operation is set aside, this message is followed by the LDIF of the operation.

All non-LDIF information is prefixed with the comment character # so that the error log can be run through **ldapmodify** to synchronize the two servers.

Following is an example in which a replication error condition is logged but no set-aside of the modification is needed:

```
#(070102 03:35:47.003707): delete operation failed for cn=IBMUSER01,
  O=YOUR COMPANY to 9.57.1.198:3389, rc=32
# R004071 DN 'cn=IBMUSER01,O=YOUR COMPANY' does not exist (ldbm_process_request)
# Entry is already deleted, ignoring request.
```

There is no LDIF. Notice the third message indicates that request is being ignored.

Troubleshooting

If the replica server does not seem to be receiving updates from the replicating server (master or peer), there are several possible reasons. Check the following conditions for a possible quick fix:

- Check for messages from the replicating server.
- Verify that a replica entry for the replica server exists in the backend to be replicated in the replicating server, and was specified correctly to match with the replica server. If **cn=localhost** is used as the suffix for all replica entries for a backend, perform an **ldapsearch** with a base of **cn=localhost** and a filter of **objectClass=***. Otherwise, perform an **ldapsearch** where the search base is the suffix defined in the backend section of the LDAP server configuration file and the filter is **objectClass=replicaObject**. If more than one suffix is configured for 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 **ldapsearch**.
- Verify that the **replicaHost** value in the replica entry for that replica specifies the machine on which the replica is running.
- Check that the values listed in the replica entry for that replica match those of the replica server configuration. Specifically, the **replicaPort**, **replicaBindDN**, and **replicaCredentials** should be verified.
- Check that the **replicaUpdateTimeInterval** specified in the replica entry for that replica has been set correctly.
- Verify that the replica server is running by performing an **ldapsearch** against the replica.
- Check that the default referral specified in the LDAP server configuration file in the replica server points to the replicating server.
- If the replica entry **replicaUseSSL** attribute is set to **TRUE**, verify the **replicaPort** attribute is set to the SSL port configured on the replica server. Verify the **sslKeyRingFile**, and **sslKeyRingFilePW** or **sslKeyRingPWStashFile** values in the LDAP server configuration file on the replica server and on the replicating server are correct.
- When adding 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 replication fails, the error log holds all errors that occurred during replication and the LDIF for the set aside replication operations.

Recovering from out-of-sync conditions

If a replica becomes out-of-sync with its replicating server for any reason, and normal replication processing is not correcting the situation, it may be necessary to reload the replica.

The following procedure should be followed to reload a replica:

1. Issue SMSG LDAPSRV MAINTMODE ON on the replicating sever and on each of the replica servers to put them into maintenance mode.
2. Using the administrator DN, unload all the replica entries (entries that describe replica servers) from the master server. Use a search command to create LDIF output containing the replica entries for each suffix in the backend.
3. Using the 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 7. Create and Customize the LDAP Configuration File (DS CONF)" in *z/VM: TCP/IP Planning and Customization* for more information on 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 **ldapmodify** 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. Using an administrator DN, run **ldapadd** to add the replica entries unloaded in step 2 back into the replicating server.
11. Issue SMSG LDAPSRV MAINTMODE OFF to take the replicating server and each replica out of maintenance mode.

Chapter 10. Alias

Alias support provides a means for an LDBM 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=IBM"
```

The alias entry points to the actual LDAP department:

```
"ou=DEPTC8NG,ou=Poughkeepsie,o=IBM_US,o=IBM"
```

This provides easier access to the entries of the LDAP developers, using public names such as:

```
"cn=kmorg,ou=LDAPZOS,o=IBM"
```

This name is dereferenced during search to:

```
"cn=kmorg,ou=DeptC8NG,ou=Poughkeepsie,o=IBM_US,o=IBM
```

and the information for that entry is returned.

Impact of aliasing on search performance

Usage of aliases in a directory can cause a large increase in the amount of processing that takes place during search, even if no alias entries are actually involved in the particular search that was requested. To minimize the impact to search performance:

- Do not add aliases to the directory if they are not needed. There is no impact on search if there are no aliases in the directory.
- Only perform a search with dereferencing when aliases are involved in the search. Again, the impact on search is avoided if no dereferencing is requested.

  Note: The search request from the LDAP client specifies whether to do dereferencing. The default value for dereferencing varies between different LDAP clients. If the default is to do dereferencing (this is the case with some Java™ clients), make sure to specifically reset this value to do no dereferencing when you issue search requests for which you do not want to do dereferencing.
- If you do want to use aliases in a directory, use them efficiently to minimize the number of alias entries. For example, use an alias entry for the root of a subtree (such as the alias for a department entry in the example above) rather than creating an alias entry for each individual entry within the subtree.

Alias entry

An alias entry contains:

- one of two object classes:
  - **aliasObject** - AUXILIARY object class
  - **alias** - STRUCTURAL object class.

    Note: This requires an object class such as **extensibleObject** to allow the naming attributes for the entry.
- **aliasedObjectName** attribute
  - its value is the distinguished name that the alias points to

These object classes and attributes are always part of the LDAP server schema.

Below is an example of an alias entry:

```
dn: ou=LDAPZOS,o=IBM
objectclass: organizationalUnit
objectclass: aliasObject
ou: LDAPZOS
aliasedobjectname: ou=DeptC8NG,ou=Poughkeepsie,o=IBM_US,o=IBM
```

or

```
dn: ou=LDAPZOS,o=IBM
objectclass: alias
objectclass: extensibleobject
ou: LDAPZOS
aliasedobjectname: ou=DeptC8NG,ou=Poughkeepsie,o=IBM_US,o=IBM
```

Alias entry rules

An alias is a directory entry containing either the **alias** structural object class or the **aliasObject** auxiliary object class. Both of these object classes require the **aliasedObjectName** attribute (the **aliasedEntryName** alternate name can also be used). The **extensibleObject** object class should also be specified if the **alias** object class is used in order to add the RDN attributes for the alias entry.

An alias entry must be a leaf entry. This means that no ancestor of an entry can be an alias entry. In addition, an alias entry cannot also be a referral entry. A suffix entry can be an alias entry. In this case, the suffix will have no entries below it.

The value of the **aliasedObjectName** attribute does not have to be an existing entry. However, an error will be returned when dereferencing the alias if the value of the **aliasedObjectName** attribute does not refer to an entry in the same backend as the alias entry. The value cannot be the distinguished name of the alias entry; in other words, an alias entry cannot dereference to itself.

Dereferencing an alias

All or part of a distinguished name (DN) can be an alias. Dereferencing a DN consists of the systematic replacement of an alias within the DN by the value of the **aliasedObjectName** attribute of the alias. This creates a new DN that must then be checked to see if it contains an alias that needs to be dereferenced. This process continues until the final dereferenced DN contains no alias within its name. An error will be returned if a circular chain is detected, that is, when a particular alias entry is encountered more than once. The final dereferenced DN must be the DN of an entry in the same backend as the original DN. This entry must either exist or be under a referral entry.

Alias dereferencing is performed only during search operations. Alias entries are not dereferenced for any other LDAP operation.

Aliases are not dereferenced when performing a null-based subtree search since all entries in all LDBM 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 requestor might not have any permissions to those alias entries.

Dereferencing during search

Dereference options

A flag value controls what alias dereferencing will be done during a search operation. This flag is sent by the client on the search request. The flag can have one of four values:

LDAP_DEREF_NEVER (0)
do not dereference any alias entries. Alias entries encountered during the search operation are processed as ’normal’ entries and are returned if they match the search filter.

LDAP_DEREF_SEARCHING (1)
dereference alias entries within the scope of the search but do not dereference the search base entry (if it contains an alias). The search base is processed as a ’normal’ entry (even if it is an alias entry) and is returned if it matches the search filter and is in the search scope.

LDAP_DEREF_FINDING (2)
dereference the search base entry (if it contains an alias) but do not dereference any other alias entries within the search scope. Alias entries within the search scope of the 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.

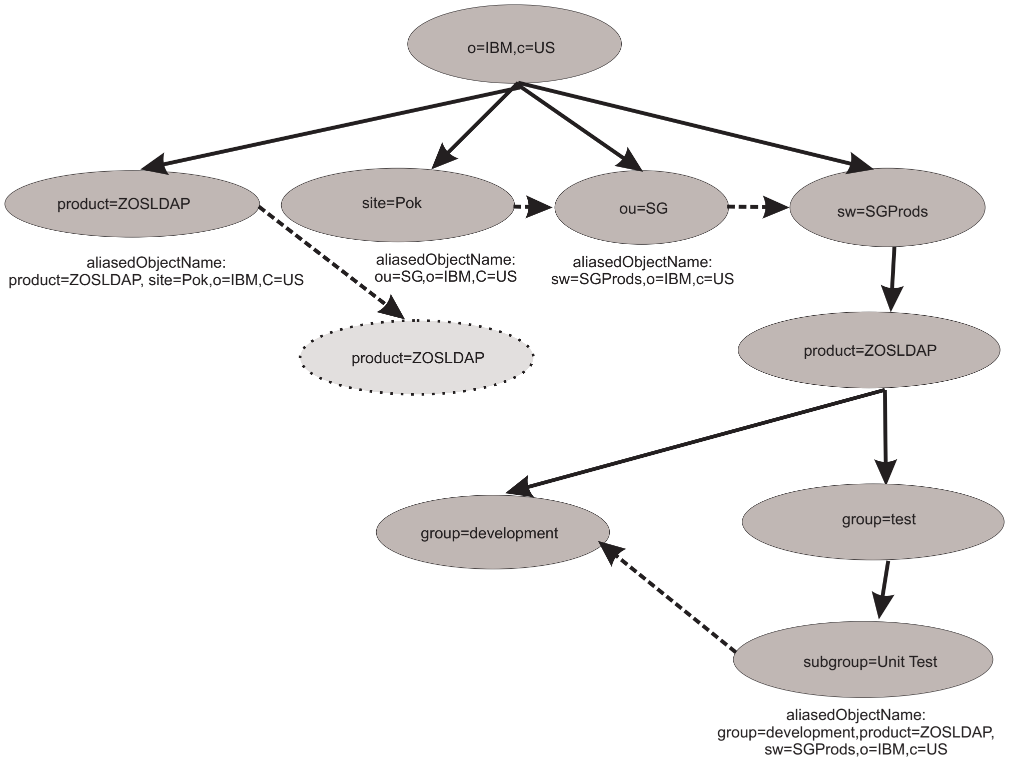


Figure 36. Alias example

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 11. 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 entry, to the LDAP server schema entry (`cn=schema`), or to an object controlled by an application (for example, a RACF user, group, or user-group connection 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/off. The **changeLogMaxEntries** and **changeLogMaxAge** configuration options determine when removal of old change log entries takes place. See "Configuring the LDAP Server" in *z/VM: TCP/IP Planning and Customization* for more information. If none of these configuration options is specified in the GDBM section, the default is to start change logging with no limits on the size of the change log.

The **changeLoggingParticipant** configuration option can be used to specify if an LDBM backend wants change log entries to be created for changes to entries in the LDBM 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 or GDBM backend, the default is to create change log entries for changes to that LDBM 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 an ACL that allows only the administrator to access the change log. The ACL is propagated to the change log entries. The user needs to use an LDAP modify operation to change this ACL to an appropriate ACL for his usage of the change log. 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, multiserver, persistentSearch, readOnly, sizeLimit**, and **timeLimit** are options can also be specified in the GDBM configuration section. The **changeLogging, changeLoggingParticipant, changeLogMaxAge**, and **changeLogMaxEntries** configuration options can be specified to control change logging activity. See "Configuring the LDAP Server" in *z/VM: TCP/IP Planning and Customization* for more information on these options.

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, or connection:

- 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, or connection. SDBM is also needed to retrieve the RACF user's new password or other changed fields.
- 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 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, 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, or user-group connection 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 and schema changes

If change logging is activated, each add, modify, delete, or modify DN operation of an entry in any LDBM backend or modify of the LDAP server schema entry results in the creation of a change log entry, with the exception of the following:

- 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 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 userid and/or groupid 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 and the LDAP server schema. However, the values for the **userPassword, secretKey**, and **replicaCredentials** 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 is changed, and contains either `*comeAndGetIt*` or `*NoEnvelope*`, for example:

```
replace: racfPassword
racfPassword: *ComeAndGetIt*
-
```

newRDN
the new RDN specified in an LDBM modify DN operation

deleteOldRdn
a boolean indicating if the old RDN was deleted in an LDBM 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 only the administrator to access 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. See Default ACLs with LDBM for more information.

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 value of the **userpassword**, **secretkey**, **replicacredentials**, and **racfpassword** attributes is replaced by `*ComeAndGetIt*`. You can use a search command to retrieve the password. For a RACF password, see Chapter 5, "Accessing RACF information," on page 59 for more information.

Unloading and loading the change log

The unload utility (**ds2ldif**) cannot be used to unload the contents of the change log. You should use the search operation to do this. Change log entries cannot be loaded into the change log. 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.
   b. If you plan to log changes to RACF users, groups, and user-group connections, you must also:

      Add the SDBM backend section. Following is an example:
      ```
      database sdbm GLDBSD31
      suffix cn=myRacf
      ```
      Enable the PC Callable support (used by RACF to add change log entries to the LDAP server) by specifying the following option in the global section of the configuration file:
      ```
      listen ldap://:pc
      ```
   c. If you do not want to log changes to entries in an LDBM backend or to the LDAP server schema entry, add the following option to the LDBM or GDBM backend section (the GDBM backend controls change logging for the schema entry):
      ```
      changeLoggingParticipant off
      ```
2. If you plan to log changes to RACF users, groups, and connections, 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, or connection, or to an LDBM 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 client access to the change log to the LDAP administrator.

For example, to give read access to the change log to RACF user CLREADER, create an ldif file, `cl.ldif`, similar to the following:

```
dn: cn=changelog
changetype: modify
add: aclentry
aclentry:access-id:racfid=clreader,profiletype=user,cn=myRacf:normal:rsc:
 sensitive:rsc:critical:rsc:system:rsc
-
```

You should then modify the change log ACL by issuing a modify command similar to the following:

```
ldapmodify  -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:

```
ldapsearch -h ldaphost -p ldapport  -D adminDn
 -w adminPw  -b "cn=changelog" "objectclass=*"
```

   Part of the output from this search would look like:

```
cn=changelog
objectclass=top
objectclass=container
cn=changelog

CHANGENUMBER=1,CN=CHANGELOG
objectclass=CHANGELOGENTRY
objectclass=IBM-CHANGELOG
objectclass=TOP
changenumber=1
targetdn=RACFID=U2,PROFILETYPE=USER,cn=myRacf
changetime=20030611204814.257756Z
changetype=MODIFY
changes=replace: racfPassword
racfPassword: *ComeAndGetIt*
-

ibm-changeinitiatorsname=RACFID=SUSET3,PROFILETYPE=USER,cn=myRacf
```

7. If the **changes** attribute of a change log entry contains any of the following lines:

```
racfPassword: *NoEnvelope*
racfPassword: *ComeAndGetIt*
userpassword: *ComeAndGetIt*
replicacredentials: *ComeAndGetIt*
secretkey: *ComeAndGetIt*
```

   then a password in the RACF user profile, LDBM 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 150 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:

```
ldapsearch -h ldaphost -p ldapport  -D adminDn
 -w adminPw -s base -b "" "objectclass=*"
```

   Part of the output from this search would look like:

```
changelog=CN=CHANGELOG
firstchangenumber=1
lastchangenumber=202
```

Note: The LDAP server occasionally skips one or more change numbers, so it cannot be assumed that there is a change log entry for every number between 1 and 202. In addition, skips are created if you delete a change log entry that does not have the lowest number. Change numbers that are generated by the LDAP server are not guaranteed to be consecutive, but will always increase.

Chapter 12. 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 may contain entries that do not fall within any of the suffixes in this LDAP server. If the target of an operation is not at or below any suffix defined in the LDAP server, the LDAP server returns the default referral to the client.

Also described in This topic is how to associate multiple servers using referrals and an example of associating a set of servers through referrals and replication (see Chapter 9, "Replication").

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.

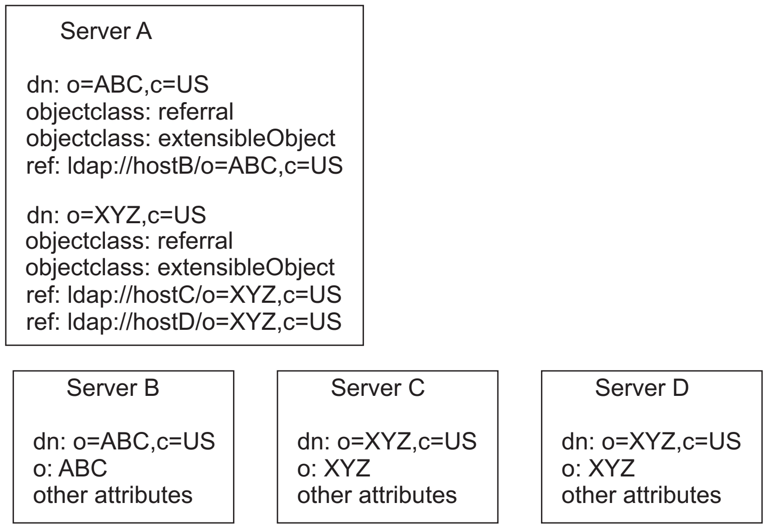


Figure 37. Example using ref attribute

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 which do not hold the needed data, servers can pass back referral URLs which indicate one or more other servers to contact. The clients can then request the information from the referenced server. The z/OS client API, by default, chases referrals returned from servers. However, client applications can suppress referral chasing through the **ldap_set_option()** API. In this case, the application retrieves the referral from the LDAP client and processes it within the application. This option's scope is the LDAP handle, so a client could open multiple connections to one or more servers, some of which would chase referrals automatically, and some of which would not.

Servers present the referral URLs differently depending on the LDAP protocol version being used by the client. Referrals are presented to LDAP Version 2 clients in the error string, as the protocol does not provide a specific mechanism for indicating referrals. In LDAP Version 3, protocol elements are specifically defined to allow servers to present referral information to clients.

Using LDAP Version 2 referrals

Referrals are not supported by the LDAP Version 2 protocol. In order to provide referral information to LDAP Version 2 clients, the referral information is returned as part of the error string in the result message. Since clients do not generally examine the error string for results indicating **LDAP_SUCCESS**, the LDAP server returns **LDAP_PARTIAL_RESULTS** instead of **LDAP_SUCCESS** if referral information is present in the error string. Referral information may be present for any result other than **LDAP_SUCCESS**.

The referral information in the error string is returned as follows, where '\n' indicates a newline character:

```
Referral:\n
ldap://hostname:port/DN\n
...
```

where `Referral:` is followed by a new line character (`\n`) and `ldap://hostname:port/DN\n` is an LDAP URL followed by a new line character. The ellipses (`...`) indicate a list of multiple referrals; that is, more LDAP URLs followed by new line characters.

Limitations with LDAP Version 2 referrals

Multiple referrals are only presented for partial search results when it is necessary to contact more than one additional server to complete the entire request. This would indicate that multiple referral entries were found in the referencing server that matched the search criteria. If chasing referrals, the client contacts every server presented in the list to continue the search request. For referral entries that have multi-valued **ref** attributes, the server sends only one of the LDAP URLs to a client using LDAP Version 2 protocol. This is because there is no provision for distinguishing between equivalent servers to contact (as indicated by multi-valued **ref** attributes) and multiple servers which must be contacted to complete a search request.

A second limitation of referrals in LDAP Version 2 is that operations can sometimes be ambiguous in their intent regarding whether the operation was targeted for "real" entries in the namespace, as opposed to the referral entries themselves. For searches, referral entries are only presented as referrals, since the usual intent of a search is to look at the real entries in the namespace. Server administrators must therefore use other means to examine existing referral entries, such as examining the database, or reviewing **ds2ldif** output. For update operations, default referrals for upward references are presented as referrals, so that read-only replica servers can forward update operations to the master replica. However, subordinate references indicated by a referral entry are not followed for update operations, rather they operate on the referral entry itself. This is necessary to allow an administrator the ability to delete or modify existing referral entries. Erroneous changes caused by misdirected update operations are generally avoided through access protection and schema rules.

Using LDAP Version 3 referrals

In LDAP Version 3, referrals are defined as part of the protocol. The LDAP Version 2 limitations mentioned above are overcome by elements of the protocol and extensions to the protocol. There are two methods of passing back referral information in the LDAP Version 3 protocol: referrals and search continuation references.

If the target of a request is a referral entry or is below a referral entry, or if the target does not fall within any of the suffixes in the LDAP server and a default referral is configured, then a result code of **LDAP_REFERRAL** is presented by the server to indicate that the contacted server does not hold the target entry of the request. The referral field is present in the result message and indicates another server (or set of servers) to contact. Referrals can be returned in response to any operation request except unbind and abandon which do not have responses. When multiple URLs are present in a given referral response, each one must be equally capable of being used to progress the operation.

If the target of a search is found in the directory but a referral entry is encountered during the rest of a one-level or subtree search, a referral is not returned. Instead, one or more search continuation references are returned. Search continuation references are intermixed with returned search entries. Each one contains a URL to another server (or set of servers) to contact, and represents an unexplored subtree of the namespace which potentially satisfies the search criteria. When multiple URLs are present in a given search continuation reference, each one must be equally capable of being used to progress the operation.

As mentioned earlier, the other limitation in LDAP Version 2 referral processing is related to the inability of a client to specify whether a request was targeted for a normal entry or a referral entry. For LDAP Version 3, this difficulty is overcome with a protocol extension in the form of the **manageDsaIT** control. (Appendix B, “Supported server controls” 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 operations utilities support the **-M** option to indicate that the requestor is managing the namespace, and therefore wishes to examine and manipulate referral entries as if they were normal entries. See *z/VM: TCP/IP User’s Guide* for more information. An exception to the processing described above is that referral entries are always treated as normal entries during the second phase of a persistent search, even if the **manageDsaIT** control is not specified on the persistent search request. See PersistentSearch 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 operation utilities like **ldapsearch** and **ldapmodify** use the bind DN and password specified on the utility invocation, both when binding to the original target server and also when chasing referrals to other servers. If you want the LDAP operation utilities to automatically chase referrals across servers, then the same bind DN and password must be accepted on each of the servers connected by referrals.

If you use an approach where there are no common bind identities, then your applications will either be limited to unauthenticated access or they will require the ability to bind appropriately to each server when chasing referrals.

Consider the following approaches:

1. Use unauthenticated access for reading information to avoid the need to bind with a common identity. This makes sense if the data in the directory is general reference information that does not need to be protected.
2. Establish an 'authentication' backend for identity information that is the same on each server. This could be an SDBM backend, where the common authentication identities are in RACF, or 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 administrator DN to access the entries, configure the administrator DN and password identically in each of the referral servers.

Example: associating servers through referrals and replication

Following are the steps involved in distributing the namespace using referrals.

1. Plan your namespace hierarchy.

   ```
   country - US
   company - IBM, Lotus
   organizationalUnit - IBM Austin, IBM Endicott, IBM HQ
   ```

2. Set up multiple servers, each containing portions of the namespace.

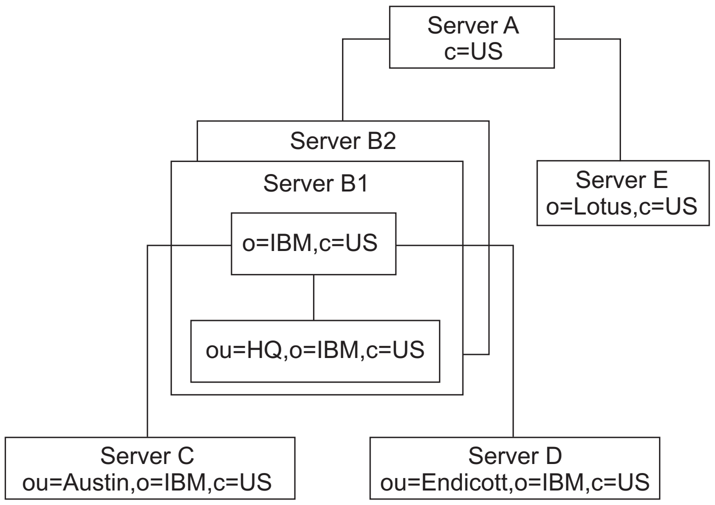


Figure 38. Setting up the servers

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.

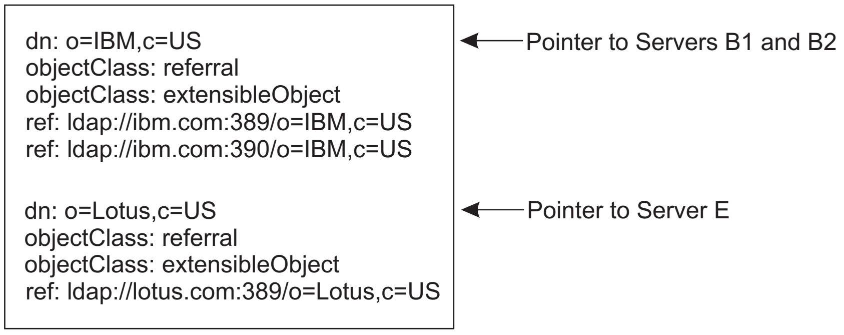
Server B2
: A replica of Server B1.

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
```

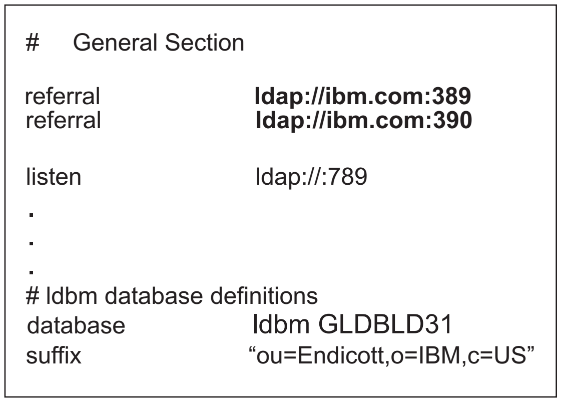
Pointer to Servers B1 and B2
Pointer to Server E

Figure 39. 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.

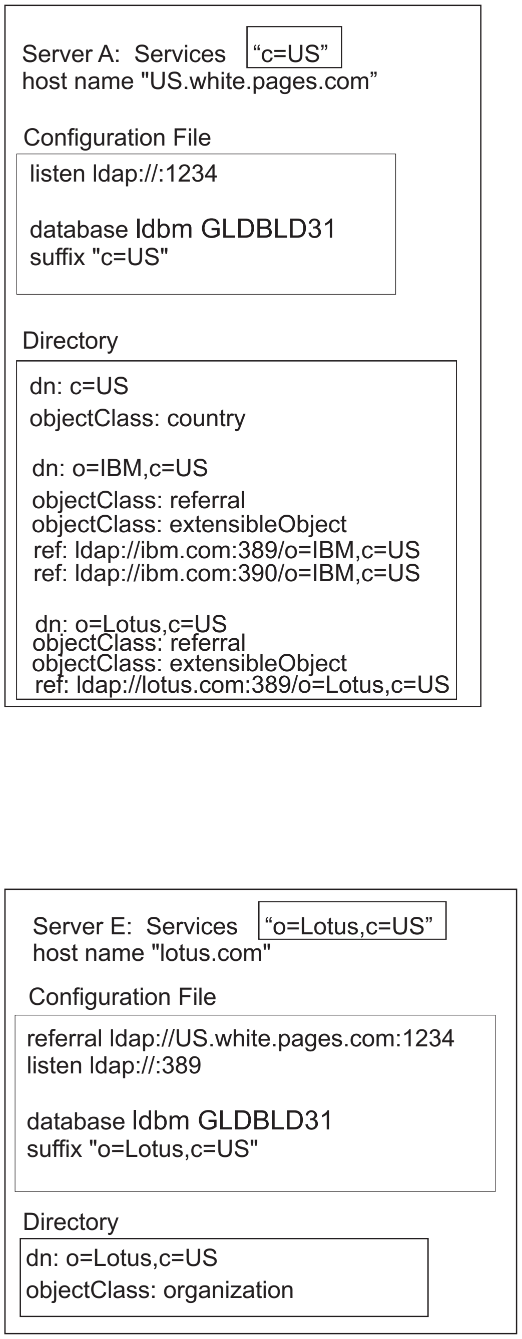


```
#     General Section

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 40. Server D configuration file

5. Putting it all together.

   Figure 41, Figure 42, Figure 43, and Figure 44 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
Directory

```
listen ldap://:1234

database ldbm GLDBLD31
suffix "c=US"
```

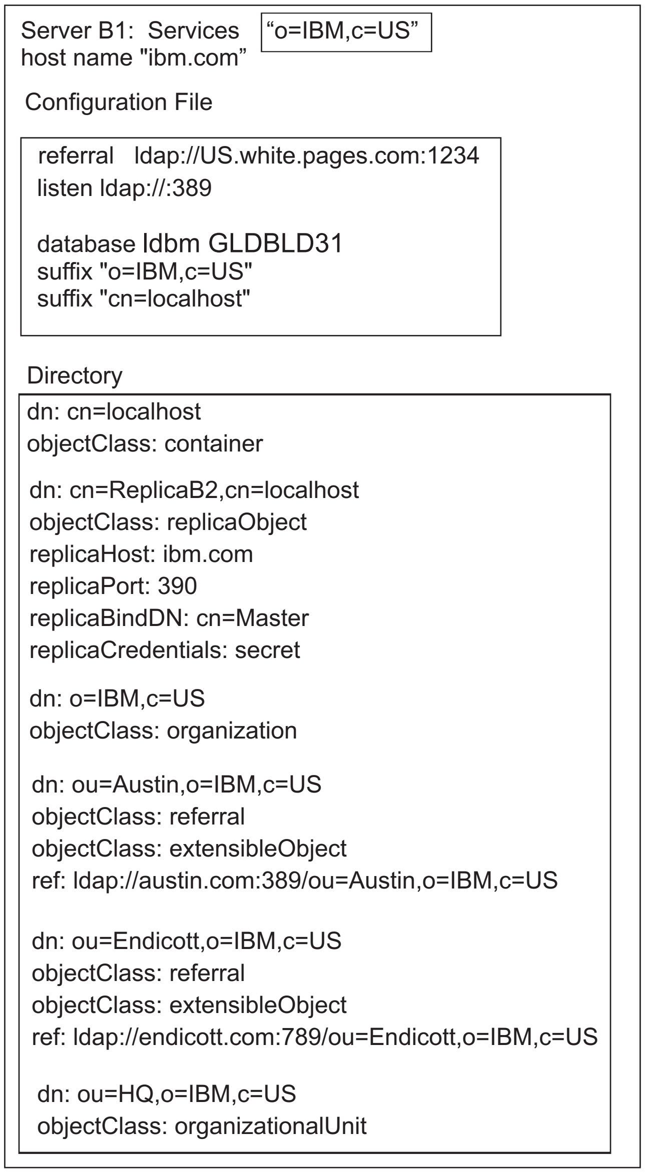
```
dn: c=US
objectClass: country

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
Directory

```
referral ldap://US.white.pages.com:1234
listen ldap://:389

database ldbm GLDBLD31
suffix "o=Lotus,c=US"
```

```
dn: o=Lotus,c=US
objectClass: organization
```

Figure 41. Referral example summary (servers A and E)


Server B1: Services “o=IBM,c=US”
host name "ibm.com”

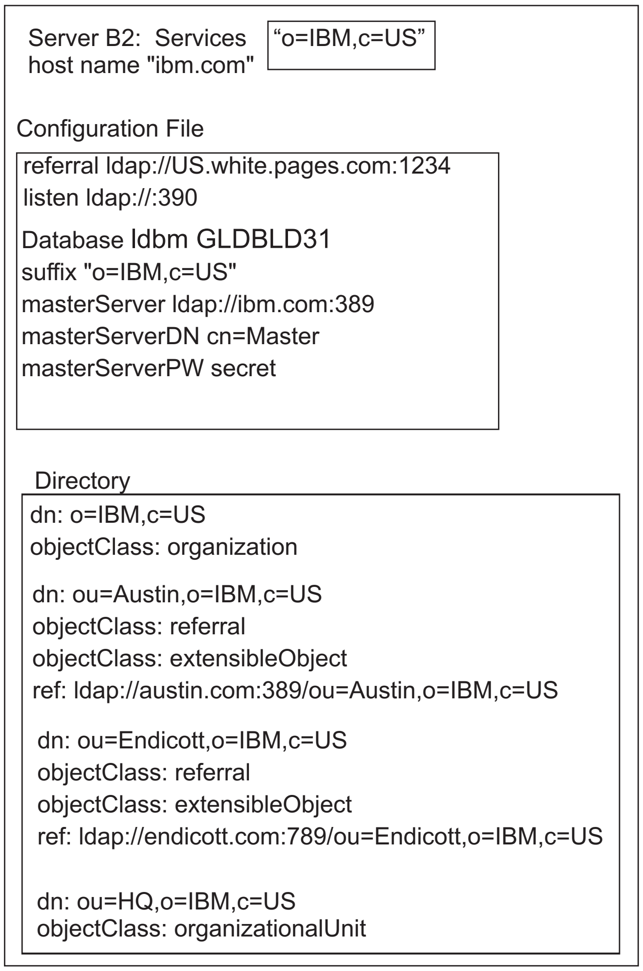
Configuration File

```
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
```

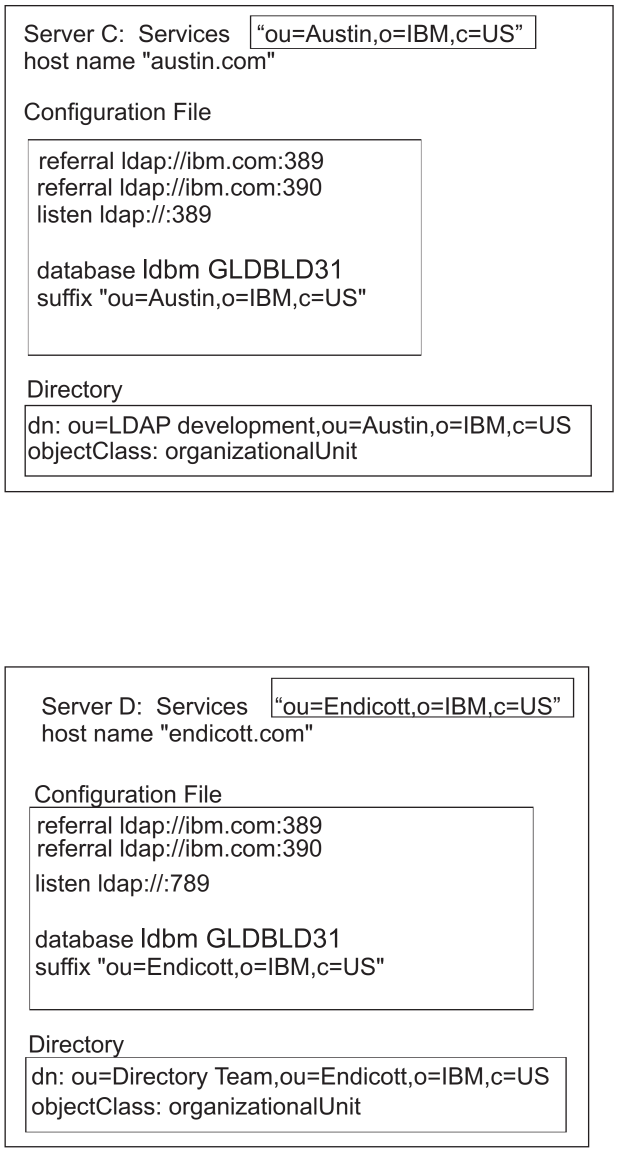
Figure 42. Referral example summary (server B1)



```
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
```

```
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 43. 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 44. Referral example summary (servers C and D)

Chapter 13. 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. See "Creating the DS CONF File" in *z/VM: TCP/IP Planning and Customization* for configuration file options.

```
# 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 45. 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 46. 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 47. 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 the **ldapadd** and **ldapmodify** tools 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) 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.

While there are a number of manual steps to perform, there is a small consolation that the steps at different locations are not interleaved. All work can be done at the source location and then all work can be performed at the target (replica) location.

Resynching the replica and master servers

If it is noticed that a replica's contents are out of sync with the information at the master server, the information can be resynched by following the steps shown in Recovering from out-of-sync conditions.

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 9, "Replication" 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 14. 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 the LDAP Version 2 protocol
- 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 the **ldapsearch** utility to request a base search of the root DSE and shows sample output for the search:

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

Following is an example of the information that the LDAP server 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.

```
supportedcontrol=2.16.840.1.113730.3.4.2
supportedcontrol=1.3.18.0.2.10.2
supportedcontrol=1.3.18.0.2.10.10
supportedcontrol=1.3.18.0.2.10.11
supportedcontrol=1.3.18.0.2.10.20
supportedcontrol=2.16.840.1.113730.3.4.3
supportedcontrol=1.3.18.0.2.10.19
supportedcontrol=1.3.18.0.2.10.6
supportedextension=1.3.6.1.4.1.1466.20037
supportedextension=1.3.18.0.2.12.8
supportedextension=1.3.18.0.2.12.7
supportedextension=1.3.18.0.2.12.48
supportedextension=1.3.18.0.2.12.62
namingcontexts=cn=myRACF
namingcontexts=CN=CHANGELOG
namingcontexts=o=IBM,c=US
```

```
namingcontexts=secAuthority=Default
ibm-supportedcapabilities=1.3.18.0.2.32.19
ibm-supportedcapabilities=1.3.18.0.2.32.3
ibm-supportedcapabilities=1.3.18.0.2.32.31
ibm-supportedcapabilities=1.3.18.0.2.32.7
ibm-supportedcapabilities=1.3.18.0.2.32.33
ibm-supportedcapabilities=1.3.18.0.2.32.34
ibm-supportedcapabilities=1.3.18.0.2.32.30
ibm-supportedcapabilities=1.3.18.0.2.32.28
ibm-supportedcapabilities=1.3.18.0.2.32.24
ibm-supportedcapabilities=1.3.18.0.2.32.26
ibm-enabledcapabilities=1.3.18.0.2.32.3
ibm-enabledcapabilities=1.3.18.0.2.32.7
ibm-enabledcapabilities=1.3.18.0.2.32.33
ibm-enabledcapabilities=1.3.18.0.2.32.34
ibm-enabledcapabilities=1.3.18.0.2.32.31
ibm-enabledcapabilities=1.3.18.0.2.32.28
ibm-enabledcapabilities=1.3.18.0.2.32.24
ibm-enabledcapabilities=1.3.18.0.2.32.26
subschemasubentry=cn=schema
supportedsaslmechanisms=EXTERNAL
supportedsaslmechanisms=CRAM-MD5
supportedsaslmechanisms=DIGEST-MD5
supportedldapversion=2
supportedldapversion=3
vendorname=International Business Machines (IBM)
vendorversion=z/VM V6R1
ibmdirectoryversion=z/VM V6R1
ibm-sasldigestrealmname=MYHOST.IBM.COM
altserver=ldap://host2.ibm.com:999
ref=ldap://hostk.ibm.com:391
changelog=cn=changelog
firstchangenumber=24213
lastchangenumber=24322
```

Following are Object Identifiers (OIDs) for supported and enabled capabilities:

Short name	Description	OID assigned
Entry UUID	Identifies that this server supports the **ibm-entryuuid** operational attribute.	1.3.18.0.2.32.3
System Restricted ACL Support	Indicates that the server supports specification and evaluation of ACLs on system and restricted attributes.	1.3.18.0.2.32.7
Max Age ChangeLog Entries	Specifies that the server is capable of retaining changelog entries based on age.	1.3.18.0.2.32.19
Monitor Operation Counts	The server provides new monitor operation counts for initiated and completed operation types.	1.3.18.0.2.32.24
Null-based subtree search	Indicates that the server supports null-based subtree search operations, which search all the LDBM entries in the server.	1.3.18.0.2.32.26
TLS Capabilities	Specifies that the server is capable of performing Transport Layer Security (TLS).	1.3.18.0.2.32.28

Short name	Description	OID assigned
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 a search on the **ibm-allGroups** operational attribute.	1.3.18.0.2.32.31
Modify DN (subtree move)	Indicates that a subtree can be moved to another subtree, within a backend. This move uses a new superior. It can also use a new RDN.	1.3.18.0.2.32.33
Modify DN (subtree rename)	Indicates that a subtree can be renamed. The DN of each entry under the subtree will also be changed. This rename uses a new RDN but not a new superior.	1.3.18.0.2.32.34

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 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 return referrals to the client. Any filter can be specified for the subtree search.

A root DSE subtree is implemented as a series of searches to each LDBM suffix. These individual searches are each limited by the **timelimit** and **sizelimit** options specified in the LDAP server configuration file. 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. See the descriptions of the **sizelimit** and **timelimit** options in “Step 7. Create and Customize the LDAP Configuration File (DS CONF)” in *z/VM: TCP/IP Planning and Customization* for more information. Each individual LDBM search is subject to the normal LDBM access control checking.

The following example uses the **ldapsearch** utility to request a subtree search of the root DSE for entries that have a `cn` value that begins with `ken` and shows sample output for the search.

```
ldapsearch -h ldaphost -p ldapport -D binddn -w bindpw -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.

CRAM-MD5 authentication support

CRAM-MD5 authentication is supported on the IBM Tivoli® Directory Server client utilities on other platforms, such as AIX®, Windows®, and Linux®. 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 z/VM LDAP server.

In order to perform a CRAM-MD5 authentication bind with the IBM Tivoli Directory Server client utilities on other platforms to the z/VM 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 the LDAP Version 2 protocol

The LDAP Version 3 Protocol allows UTF-8 attribute values outside of the IA5 character set to be stored in the directory. This information must be able to be returned in some format to LDAP Version 2 clients. An additional LDAP server configuration file option, **sendV3stringsoverV2as**, which has the possible values **ISO8859-1** or **UTF-8**, can be used to indicate which format to use when sending this information over the Version 2 protocol.

Note: Different clients treat non-IA5 data differently over the Version 2 protocol. 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.

Reason codes

The **LDAPResult** construct is used by the LDAP protocol to return success or failure indications from servers to clients. This construct contains an error message field. Servers can optionally provide "human-readable" diagnostic information in this field. Depending on the location in the LDAP server where errors are detected, error messages generated may have the following format:

```
R<numeric digits> <diagnostic information> <traceback information>
```

where:

numeric digits
Represents a specific reason code.

diagnostic information
Provides details about the reason for the failure.

traceback information
Is of the form (*file_identification*) and will assist you in diagnosing application or configuration problems.

Note the following regarding this error information:

- It is intended to be "human-readable" to assist in identifying problems detected by the server.
- It is not translated (English text only).
- It is not intended to be used as an application programming interface (API).
- Data returned may be changed by service or new releases of the product. (Again, it is not intended to be an API.)
- The reason code returned for a particular error can change and the reason code text can change.

Following is the current list of reason codes and associated diagnostic information returned by the LDAP server.

R000001	**Unable to allocate storage**
R000004	**Internal server error encountered**
R000005	**Unable to translate value for attribute '*name*' from *source_codepage* to *target_codepage***
R000100	**The password has expired**
R000101	**The new password is not valid**
R000102	**The user id has been revoked**
R000104	**The password is not correct or the user id is not completely defined (missing password or uid)**
R000105	**A bind argument is not valid**
R000114	**The realm portion of the value of attribute '*name*' is not the RACF default realm**
R000115	**There is no RACF default realm**
R000116	**Cannot specify a value when deleting attribute '*name*'**
R000117	**Cannot delete attribute '*name*'**
R000118	**Cannot replace attribute '*name*'**
R000119	**Cannot add or replace attribute '*name*'**
R000120	**Cannot specify multiple values for attribute '*name*'**
R000121	**Value for attribute '*name*' must be same as value for DN**
R000122	**The value for attribute '*name*' must be the DN of a user**
R000123	**The value for attribute '*name*' must be the DN of a group**
R000124	**The value for attribute '*name*' must be the DN of a user or a group**
R000125	**Attribute '*name*' is not supported**
R000126	**Filter '*filter*' is not supported for this base**
R000127	**Filter '*filter*' contains a type without a value**

R000128 **Filter is not supported**

R000129 **Value '*value*' is not supported for filter '*filter*'**

R000131 **'*name*' is not a valid RACF DN**

R000132 **Value for attribute '*name*' cannot be more than *size* characters**

R000133 **Value for attribute '*name*' must be an integer less than *size***

R000134 **The RACF *type* command created to satisfy this request is too long, probably due to specifying a long filter or attribute value or too many attribute values**

R000135 **Cannot perform this request on a reserved SDBM DN, '*name*'**

R000137 **'*name*' is not a valid RACF DN for bind, check that the syntax is correct for a RACF user DN**

R000139 **RACF '*type*' command failed**

R000140 **Cannot parse RACF '*command*' output**

R000141 **Routine '*name*' failed, rc=*return_code***

R000142 **Cannot obtain the password of a RACF user**

R000143 **Bound user does not have the authority to perform this operation**

R000144 **Cannot specify a binary attribute in a compare operation**

R000145 **Must specify a value when deleting attribute '*name*'**

R000200 **Change log not active**

R000201 **Cannot decode *attribute* from request, rc=*return_code***

R000202 **Request did not come over PC interface**

R000203 **Value for *attribute* out of range**

R000204 **Required value for *attribute* is missing**

R000205 **Unable to convert userID (*value*) and Group (*value*) to DN, rc=*return_code***

R000206 **PC caller must be in supervisor state**

R001001 **Generalized Time value '*value*' is not valid**

R001005 **Duplicate value encountered: '*value*'**

R001008 **Value specified for attribute '*name*' does not match attribute syntax**

R001011 **COLLECTIVE keyword is not supported for attribute type '*name*'**

R001012 **Attribute type '*name*' is not defined**

R001015 **Cycle detected in superior hierarchy for '*identifier*'**

R001017 **Syntax/matching rule inconsistency for attribute type '*name*'**

R001018 **Attribute type '*name*' is obsolete**

R001024 **Abstract class '*name*' may not be a base object class**

R001025 **Multiple base structural object classes specified for '*name*'**

R001026 **No structural object class specified for '*name*'**

R001027 **Base structural object class '*name*' may not be changed**

R001029 **Entry does not contain MUST attribute '*name*'**

R001030	**Entry contains attribute '*name*' which is not allowed for object class**
R001031	**Missing left parenthesis in definition:** *definition*
R001032	**Missing right parenthesis in definition:** *definition*
R001038	**Numeric object identifier '*value*' is not valid**
R001046	**Missing closing quote for value '*value*'**
R001047	**Missing opening quote for value '*value*'**
R001048	**Missing closing brace for value '*value*'**
R001052	**Non-numeric character found in integer value '*value*'**
R001053	**Integer value of length** *size* **exceeds maximum length of** *size*
R001055	**Attribute type '*name*' is not valid for the directory schema**
R001056	**Object class '*name*' is not valid for the directory schema**
R001060	**Object class '*name*' is obsolete**
R001067	*keyword* **keyword missing in schema definition:** *definition*
R001069	**Reference attribute type not found for IBM attribute type '*name*'**
R001072	**More than one object class type keyword found in schema definition:** *definition*
R001075	**Object identifier missing in schema definition:** *definition*
R001076	*keyword* **keyword specified multiple times in schema definition:** *definition*
R001077	*keyword* **keyword not supported in schema definition:** *definition*
R001078	**Value missing for** *keyword* **keyword in schema definition:** *definition*
R001079	**Unsupported value for** *keyword* **keyword in schema definition:** *definition*
R001080	**Attribute type '*identifier*' is already defined**
R001081	**Object class '*identifier*' is already defined**
R001082	**Inappropriate** *type* **matching rule in schema definition:** *definition*
R001083	**Object class '*identifier*' is not defined**
R001084	**IBM attribute type '*identifier*' is not defined**
R001085	**IBM attribute type '*identifier*' is already defined**
R001086	**No syntax value specified for attribute type '*identifier*'**
R001087	**Attribute type '*identifier*' is in use and cannot be replaced or deleted**
R001088	**Object class '*identifier*' is in use and cannot be replaced or deleted**
R001089	**Attribute type name '*name*' is already assigned**
R001090	**Object class name '*name*' is already assigned**
R001091	**TOP object class not found in superior hierarchy for '*identifier*'**
R001092	**Unable to save directory schema**
R001093	**Schema lookup failed while resolving references for '*identifier*'**

R001094	**Attribute type '*identifier*' is referenced by '*identifier*' and cannot be deleted**
R001095	**Object class '*identifier*' is referenced by '*identifier*' and cannot be deleted**
R001096	**OID change not allowed because the new definition is not the same as the current definition**
R001097	**Attribute type '*identifier*' conflicts with existing type, cannot be replaced for migration**
R001098	**Object class '*identifier*' conflicts with existing class, cannot be replaced for migration**
R001099	**Duplicate values specified for attribute '*name*'**
R002001	**Missing equal sign in DN component '*component*'**
R002004	**Incomplete escape sequence in DN component '*component*'**
R002006	**Empty DN component is not supported**
R002007	**Incorrect syntax in aclEntry attribute value '*value*'**
R002008	**Permissions missing in aclEntry attribute value '*value*'**
R002018	**An extraneous colon was found in aclEntry attribute value '*value*'**
R002019	**An unsupported extensible filter was specified**
R002020	**A decoding error has been encountered while base64-decoding attribute '*name*'**
R002021	**An incorrectly formatted '*name*' attribute value has been encountered**
R003029	**The aclPropagate attribute requires the aclEntry attribute**
R003030	**The '*name*' attribute cannot be used in the entry distinguished name**
R003032	**The ownerPropagate attribute requires the entryOwner attribute**
R003057	**Access denied because user does not have 'add' permission for the parent entry**
R003062	**Access denied because user does not have 'write' permission for all attributes in the new entry**
R003070	**Access denied because user does not have 'write' permission for all modified attributes**
R003076	**Access denied because user does not have 'delete' permission for the entry**
R003082	**Access denied because user does not have 'write' permission for all attributes in the old name**
R003095	**Access denied because user does not have 'compare' permission for the attribute**
R003119	**Access denied because user does not have 'write' permission for all attributes in the new name**
R003125	**Access denied because user does not have 'add' permission for the new superior entry**
R003128	**Unable to realign DN attributes because user does not have 'write' permission for attributes in '*name*'**
R003129	**Realigning DN attributes would result in duplicate values for attribute '*name*' in '*name*'**
R004017	**No attributes specified for entry '*name*'**

R004019	**Entry data is missing required RDN components**
R004020	**RDN contains duplicate values for attribute ’*name*’**
R004022	**Parent not found for entry ’*name*’**
R004026	**Entry ’*name*’ not found in database**
R004028	**Search size limit exceeded**
R004031	**Search time limit exceeded**
R004035	**Attribute type ’*name*’ may not be added or modified by users**
R004038	**Operation not allowed because backend is in read-only mode**
R004041	**Entry ’*name*’ is not a leaf and may not be deleted**
R004051	**Entry ’*name*’ does not contain attribute ’*name*’**
R004054	**Invalid UTF-8 character found in string value ’*value*’**
R004060	**Entry does not contain a password**
R004062	**Credentials are not valid**
R004071	**DN ’*name*’ does not exist**
R004073	**Entry is not a leaf and cannot be modified to be a referral entry**
R004077	**DN ’*name*’ already exists**
R004083	**New superior is not allowed for an LDAP V2 request**
R004086	**Entry ’*name*’ already contains attribute ’*name*’ with value ’*value*’**
R004091	**Non-IA5 data received for an LDAP V2 request**
R004096	**Entry ’*name*’ does not contain attribute ’*name*’ with value ’*value*’**
R004098	**Filtering on non-textual attribute ’*name*’ is not allowed**
R004099	**Parent of new entry ’*name*’ is a referral entry**
R004107	**The __passwd function failed; not loaded from a program controlled library**
R004108	**Native user ID *name* is either not defined or no UID is present in the OMVS segment**
R004109	**The password has expired**
R004110	**The user id has been revoked**
R004111	**The password is not correct**
R004112	**A bind argument is not valid**
R004113	**Native authentication cannot be performed when multiple uid values exist**
R004114	**The modify-delete of the old password must occur before the modify-add of the new password**
R004115	**More than one password cannot be specified for a native authentication password update**
R004116	**Password change not allowed because native updates are not enabled**
R004117	**Native authentication replace is not allowed**
R004118	**Native user ID ’*name*’ is either not defined or no UID is present in the OMVS segment**

R004119	**A modify-add of the new password must follow the modify-delete of the old password**
R004120	**The userPassword attribute cannot be added because the entry uses native authentication**
R004121	**Entry is using native authentication but without a native userid**
R004128	**Native authentication password change failed: The new password is not valid, or does not meet requirements**
R004129	**New superior '*name*' does not exist**
R004130	**Time limit exceeded for Modify DN operation**
R004132	**The new superior DN must exist in the same backend**
R004133	**The new superior DN is located in the subtree to be moved**
R004141	**New RDN '*name*' is not valid**
R004145	**The new superior may not be a referral or alias object**
R004153	**Parent of new entry '*name*' is an alias entry**
R004154	**Entry is not a leaf and cannot be modified to be an alias entry**
R004155	**Alias entry '*name*' points to itself**
R004158	**Cycle detected while dereferencing alias '*name*'**
R004159	**Dereferencing '*name*' failed because the resulting DN does not exist in this backend**
R004160	**Entry '*name*' cannot be both an alias and a referral**
R004161	**Persistent search terminated because search base entry has been deleted**
R004163	**Dynamic group URL '*url*' is not valid**
R004164	**An unsupported value '*value*' is specified for attribute 'ref'**
R004166	**The LDAP server is shutting down**
R004176	**The __passwd() function failed with error** *error_code*
R005001	**Requested operation is not supported by the GDBM backend**
R005002	**Only the base change log entry can be modified**
R005003	**The base change log entry cannot be deleted**
R005004	**Only the aclEntry and entryOwner attributes can be modified**
R006001	**LDAP Client API** *api_name* **has returned an error code=***return_code* **with an error message='***error_string*'
R006003	**A decoding error has been encountered while decoding attibute(s):** *attr_type*, **rc=***return_code*
R006004	**An encoding error** *return_code* **has been encountered while encoding response**
R006006	**Unsupported or inappropriate critical control '***identifier*'
R006009	**The extended operation request with OID=***oid* **requires the critical control with OID=***oid*
R006010	**Unsupported extended operation** '*identifier*'
R006011	**The extended operation request with OID=***oid* **does not support the critical control with OID=***oid*

R006023	**Required field (*name*) missing**
R006024	**Connection to server (*url*) failed**
R006025	**Incorrect ldapURL specified (*url*)**
R006026	**ldap_search failed rc=*return_code***
R006027	**Unsupported authorization type=*type***
R006028	**Expected attribute *name* missing from entry**
R006029	**Empty sequence in extended operation request *name***
R006050	**Extended operation request does not have an object identifier**
R006051	**The *type* backend is not defined**
R006052	**Persistent search is allowed only when bound as the LDAP administrator**
R006053	**Persistent search must specify LDAP_DEREF_NEVER or LDAP_DEREF_FINDING**
R006054	**Persistent search is not allowed using the Program Call interface**
R006055	**Persistent search is not allowed with paged or sorted results**
R006056	**Persistent search is not supported by the backend**
R006060	**Unload extended operation is allowed only when bound as the LDAP administrator**
R006061	**Unload extended operation found multiple LDBM or TDBM backends to unload**
R006062	**Unload extended operation cannot find subtree DN '*name*' to unload**
R006063	**Unload extended operation cannot find backend name '*name*' to unload**
R006064	**Unload extended operation unable to open file '*file_name*', errno=*error_code*, errstring=*error_string***
R006065	**Unload extended operation has both backend name and subtree DN specified**
R006066	**Unload extended operation cannot find any LDBM or TDBM backend in the LDAP server configuration file to unload**
R007001	**SASL authentication requires the LDAP Version 3 protocol**
R007002	**Unsupported SASL authentication method '*name*'**
R007005	**Server is not configured for client authentication**
R007006	**Client certificate is not available**
R007020	**User password is not available with native authentication**
R007027	**TLS is not supported on the connection**
R007028	**SSL/TLS is already active on the connection**
R007029	**Other operations are outstanding for the connection**
R007030	**Multiple '*name*' attributes found in DIGEST-MD5 response**
R007031	**Required '*name*' attribute not found in DIGEST-MD5 response**
R007032	**Syntax error in DIGEST-MD5 response**
R007033	**Authorization DN in DIGEST-MD5 response does not match DN associated with user name**

R007034 **BIND DN '*name*' is not the same as authentication DN '*name*'**

R007035 **The value of DIGEST-MD5 response attribute '*name*' is not valid**

R007036 **The DIGEST-MD5 authorization identifier is not a distinguished name**

R007037 **DIGEST-MD5 response attribute '*name*' is not the same as the challenge value**

R007038 **Maximum DIGEST-MD5 buffer size must be at least 256 bytes**

R007047 **SASL EXTERNAL bind using the system identity requires the SDBM backend**

R007051 **DIGEST-MD5 response URL '*url*' is incorrect or cannot be verified**

R007052 **LDAP server in maintenance mode; operations restricted to adminDN, masterServerDN and peerServerDN**

R007060 **SASL bind is in progress**

R007061 **No SASL mechanism specified**

R007062 **The EXTERNAL SASL mechanism is not available for the connection**

R007063 **Client credentials may not be specified for the EXTERNAL SASL mechanism**

R007064 **Concurrent BIND requests are not supported**

R007065 **No SASL BIND credentials**

R007066 **Unable to accept GSSAPI security context: Major 0x*status*, Minor 0x*status* - *error_string***

R007067 **Unexpected security token received for GSSAPI continuation**

R007068 **Unable to wrap GSSAPI response: Major 0x*status*, Minor 0x*status* - *error_string***

R007069 **A GSSAPI authorization identity may not be specified**

R007070 **Unable to unwrap GSSAPI response: Major 0x*status*, Minor 0x*status* - *error_string***

R007071 **Requested GSSAPI security layer *number* is not supported**

R007072 **Maximum GSSAPI receive length *size* is too small**

R007073 **Unable to get GSSAPI wrap size limit: Major 0x*status*, Minor 0x*status* - *error_string***

R007074 **Unable to obtain GSSAPI source name: Major 0x*status*, Minor 0x*status* - *error_string***

R007075 **Unexpected SASL BIND credentials**

R007076 **No digest realm name is available**

R007077 **No user name specified for SASL BIND request**

R007078 **HMAC digest in SASL BIND request is not valid**

R007079 **The local Program Call interface supports just the EXTERNAL SASL mechanism**

R007080 **A bind DN has been specified without a password**

R007081 **Anonymous binds are not allowed and no bind distinguished name exists**

R007082 **An internal SSL error has been encountered**

R007083	**Authentication with a reserved bind DN is not allowed**
R008001	**LDBM backend database is disabled**
R008002	**Entry ’*name*’ contains multiple password values**
R008003	**Multiple entries contain uid ’*name*’**
R008004	**Clear password is not available**
R008005	**Nested group recursion detected for group ’*name*’**
R008006	**Dynamic group search filter ’*filter*’ is not valid**
R008008	**No base entry specified in dynamic group URL ’*url*’**
R008009	**An internal LDBM backend error has occurred**
R008010	**Subtree move is not supported by the replica servers**
R008011	**Subtree rename is not supported by the replica servers**
R008012	**New superior is not supported by the replica servers**
R008013	**DN attribute realignment is not supported by the replica servers**
R008014	**Value *value* for attribute *name* is not valid**
R008015	**Value *value* for attribute *name* is out of range**
R008016	**SSL support is not configured**
R008102	**Entry ’*name*’ contains multiple password values**
R008103	**Multiple entries contain uid ’*name*’**
R008104	**Clear password is not available**
R008105	**Nested group recursion detected for group ’*name*’**
R008106	**Dynamic group search filter ’*value*’ is not valid**
R008108	**No base entry specified in dynamic group URL ’*url*’**
R008117	**Attribute object identifier ’*identifier*’ is longer than 200 characters**
R008118	**Object class name ’*name*’ is longer than 200 characters**
R008119	**DN ’*name*’ exceeds the maximum length of *size***
R008120	**Subtree move is not supported by the replica servers**
R008121	**Subtree rename is not supported by the replica servers**
R008122	**New superior is not supported by the replica servers**
R008123	**DN attribute realignment is not supported by the replica servers**
R008124	**Changelog root must have an explicit and propagating ACL**
R010001	**Invalid character in descriptor ’*descriptor*’**
R010002	**Missing attribute type in DN component ’*component*’**
R010003	**Missing attribute value in DN component ’*component*’**
R010004	**No equality matching rule for DN attribute ’*attribute*’**

R010005 **No matching rule defined for string value ’*value*’**

R010006 **UTC Time value ’*value*’ is not valid**

R010007 **Invalid IA5 character found in string value ’*value*’**

R010008 **Bit string value ’*value*’ is not valid**

R010009 **Boolean value ’*value*’ is not valid**

R010010 **Octet string value ’*value*’ is not valid**

R010011 **Telephone number value ’*value*’ is not valid**

R010012 **UUID value ’*value*’ is not valid**

R010013 **Undefined LDAP syntax *syntax***

R010014 **Country string value ’*value*’ is not valid**

R010015 **No backend for DN ’*name*’**

R010016 **Backend initialization failed for DN ’*name*’**

R010017 ***operation* is not supported by the *type* backend**

R010018 **Search with null base DN requires either scope=base (for root DSE search) or scope=subtree (for null based subtree search)**

R010019 **Search with null base DN requires filter (objectclass=*)**

R010020 **Schema search requires scope=base**

R010021 **Schema search requires an object class presence or equality filter**

R010022 **Binary option is not supported by the *type* backend**

R010023 **LDAP protocol version 3 is required for server controls**

R010024 **Unable to decode value for control ’*identifier*’**

R010025 **No value provided for control ’*identifier*’**

R010026 **Attribute type ’*identifier*’ already specified for a sort key**

R010027 **Control ’*identifier*’ is specified multiple times**

R010028 **Critical control ’*identifier*’ cannot be processed**

R010029 **Maximum of *size* result sets has been reached**

R010030 **Unable to compute search message digest**

R010031 **Page size of zero is not valid for initial request**

R010032 **Paged search results not found**

R010033 **Continuation search request not same as initial request**

R010034 **Unknown LDAP message type *type***

R010035 **Binary attribute type ’*name*’ not allowed in DN**

R010036 **No value provided for attribute ’*name*’**

R010037 **Binary transfer is not supported for non-binary attribute type ’*name*’**

R010038 **Embedded string delimiter found in value for attribute ’*name*’**

R010039 **Incorrect ASN.1 encoding in DN component ’*component*’**

R010040 **Unsupported ASN.1 type in DN component '*component*'**

R010041 **Server control does not have an object identifier**

R010042 **Definition has no components:** *definition*

R010043 **Substring filter for attribute '*name*' has no value**

R010044 **Substring filter type *type* is used incorrectly**

R010045 *type* **filter has an empty filter set**

R010046 **No equality matching rule for attribute type '*name*'**

R010047 **The new entry DN must exist in the same backend**

R010048 **The specified permissions are not allowed for the access class in aclEntry attribute value '*value*'**

R010049 *routine* **failed with return code** *return_code*, **reason code** *reason_code*

R010050 **Label '*name*' is not defined**

R010051 **ICSF services are not available**

R010052 **Incorrect key length for label '*name*'**

R010053 **Incorrect key parity for label '*name*'**

R010054 **Encryption type *type* is not supported**

R010055 **Encryption tag '*value*' is not supported**

R010056 **Encrypted data length is not a multiple of** *number*

R010057 **Incorrect key value for label '*name*'**

R010058 **Old and new password values were not supplied**

R010060 **LDAP protocol version 3 is required for extended operations**

R010061 **Only GetDnForUserid and GetPrivileges extended operations are supported**

R010062 **Unable to communicate with cross-system group owner**

R010063 **cn=monitor search requires scope=base**

R010064 **cn=monitor search requires filter (objectclass=*)**

R010065 **Unable to write attribute type '*name*'**

R010066 **Unable to write to file '*file_name*':** *error_code*/*reason_code* **-** '*error_string*'

Chapter 15. 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 (termed PKI), public/private key pairs must be generated. In addition, X.509 certificates (which contain public keys) may need to be created, or certificates must be requested, received, and managed.

SSL uses the **gskkyman** utility to manage PKI private keys and certificates. Invoke the **gskkyman** utility with the CMS GSKKYMAN command. **gskkyman** creates, fills in, and manages a file that contains PKI private keys, certificate requests, and certificates. This file is called a **key database** and, by convention, has a file 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 Information” 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 16. Performance tuning

Overview

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 which may require tuning to accommodate the LDAP server. For example, the SDBM backend provides access to the RACF database, which 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 CPUs 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 and GDBM backends. For LDBM and file-based GDBM, this cache helps reduce processing time for searches with complex filtering. Note that the GDBM filter cache is disabled, by default.

Operations monitor

If the operations monitor is enabled, the LDAP z/VM 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 199 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 23 on page 202 for more information about the **cn=operations,cn=monitor** entry and its attributes.

LDBM performance considerations

The LDAP server LDBM backend uses the OpenExtensions 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 195.

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 which holds the data, and the competition for resources due to 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 which 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 an OpenExtensions 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.

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 20. Server statistics

currentconnections	Current number of client connections
currenttime	Current date and time on the server
livethreads	Configured number of communication threads (**commThreads**)
maxconnections	Configured maximum number of connections (**maxConnections**)
maxreachedconnections	High water mark for concurrent client connections
resets	Number of times statistics were reset
resettime	Date and time statistics were last reset
starttime	Date and time the server was started
sysmaxconnections	System defined maximum number of connections
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 on 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 21. Server and backend specific statistics

abandonsrequested	Number of abandon operations requested
abandonscompleted	Number of abandon operations completed
addsrequested	Number of add operations requested
addscompleted	Number of add operations completed
bindsrequested	Number of bind operations requested
bindscompleted	Number of bind operations completed
bytessent	Number of bytes of data sent
comparesrequested	Number of compare operations requested
comparescompleted	Number of compare operations completed
deletesrequested	Number of delete operations requested
deletescomplketed	Number of delete operations completed
entriessent	Number of search entries sent
extopsrequested	Number of extended operations requested
extopscompleted	Number of extended operations completed
modifiesrequested	Number of modify operations requested
modifiescompleted	Number of modify operations completed
modifydnsrequested	Number of modifyDn operations requested

Table 21. Server and backend specific statistics (continued)

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
searchescompleted	Number of search operations completed
unbindsrequested	Number of unbind operations requested
unbindscompleted	Number of unbind operations completed
unknownopsrequested	Number of unrecognized operations completed
unknownopscompleted	Number of unrecognized operations completed

When statistics are reset, all of the server and backend specific statistics listed above are set to zero.

Table 22. Backend specific statistics

acl_source_cache_size	Configured maximum size (in entries) of the ACL Source cache (**aclSourceCacheSize**)
acl_source_cache_current	Current size (in entries) of the ACL Source cache
acl_source_cache_hit	Number of lookups that have hit the ACL Source cache
acl_source_cache_miss	Number of lookups that have missed the ACL Source cache
acl_source_cache_percent_hit	Percent of lookups that have hit the ACL Source cache
acl_source_cache_refresh	Number of times the ACL Source cache was invalidated
acl_source_cache_refresh_avgsize	Average number of entries in the ACL Source cache at invalidation
dn_cache_size	Configured maximum size (in entries) of the DN cache (**dnCacheSize**)
dn_cache_current	Current size (in entries) of the DN cache
dn_cache_hit	Number of lookups that have hit the DN cache
dn_cache_miss	Number of lookups that have missed the DN cache
dn_cache_percent_hit	Percent of lookups that have hit the DN cache
dn_cache_refresh	Number of times the DN cache was invalidated
dn_cache_refresh_avgsize	Average number of entries in the DN cache at invalidation
dn_to_eid_cache_size	Configured maximum size (in entries) of the DN to Entry ID cache (**dnToEidCacheSize**)
dn_to_eid_cache_current	Current size (in entries) of the DN to Entry ID cache
dn_to_eid_cache_hit	Number of lookups that have hit the DN to Entry ID cache
dn_to_eid_cache_miss	Number of lookups that have missed the DN to Entry ID cache
dn_to_eid_cache_percent_hit	Percent of lookups that have hit the DN to Entry ID cache
dn_to_eid_cache_refresh	Number of times the DN to Entry ID cache was invalidated
dn_to_eid_cache_refresh_avgsize	Average number of entries in the DN to Entry ID cache at invalidation
entry_cache_size	Configured maximum size (in entries) of the Entry cache (**entryCacheSize**)
entry_cache_current	Current size (in entries) of the Entry cache
entry_cache_hit	Number of lookups that have hit the Entry cache

Table 22. Backend specific statistics (continued)

entry_cache_miss	Number of lookups that have missed the Entry cache
entry_cache_percent_hit	Percent of lookups that have hit the Entry cache
entry_cache_refresh	Number of times the Entry cache was invalidated
entry_cache_refresh_avgsize	Average number of entries in the Entry cache at invalidation
entry_owner_cache_size	Configured maximum size (in entries) of the Entry Owner cache (**entryOwnerCacheSize**)
entry_owner_cache_current	Current size (in entries) of the Entry Owner cache
entry_owner_cache_hit	Number of lookups that have hit the Entry Owner cache
entry_owner_cache_miss	Number of lookups that have missed the Entry Owner cache
entry_owner_cache_percent_hit	Percent of lookups that have hit the Entry Owner cache
entry_owner_cache_refresh	Number of times the Entry Owner cache was invalidated
entry_owner_cache_refresh_avgsize	Average number of entries in the Entry Owner cache at invalidation
filter_cache_size	Configured maximum size (in entries) of the Filter cache (**filterCacheSize**)
filter_cache_current	Current size (in entries) of the Filter cache
filter_cache_hit	Number of lookups that have hit the Filter cache
filter_cache_miss	Percent of lookups that have hit the Filter cache
filter_cache_percent_hit	Percent of lookups that have hit the Filter cache
filter_cache_refresh	Number of times the Filter cache was invalidated
filter_cache_refresh_avgsize	Average number of entries in the Filter cache at invalidation
filter_cache_bypass_limit	Configured Filter cache bypass limit (**filterCacheBypassLimit**)
namingcontext	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. An LDBM backend reports **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 due to directory update operations. When this occurs, **cache_refresh** is incremented and **cache_current** is set to zero to reflect the refreshed (empty) cache. The **cache_hit, cache_miss**, and values **cache_percent_hit** are accumulated across cache invalidation and refresh until a RESET MONITOR command is issued or the server ends.

Table 23. Operations monitor statistics

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 23. Operations monitor statistics (continued)

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:

clientIP
: Client IP address (omitted for **searchStats** search patterns).

baseDN
: Distinguished name of the base of the search, with `_v` substituted for attribute values.

attributes
: List of attributes to be returned.

scope
: **base** for base object searches, **one** for one-level searches, and **sub** for subtree searches.

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.

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`.

numOps
: Total number of times this search pattern has occurred.

avg
: Average elapsed time for each occurrence of search pattern in microseconds.

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.

maxRate
: The highest rate on this entry.

maxRateTimeStamp
: Date and time **maxRate** was last set, in ZULU time stamp format.

createTimeStamp
: Date and time this search pattern was first added, in ZULU time stamp format.

See Table 23 on page 202 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

Following is an example of a monitor search using scope=base. This returns only statistics related to the entire server:

```
ldapsearch -h ldaphost -p ldapport -b cn=monitor -s base objectclass=*
```

```
cn=monitor
version=z/VM Version 6 Release 1 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=Mon Sep 25 16:33:00.187846 2008
starttime=Mon Sep 25 15:52:21.693392 2008
resettime=Mon 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, 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.

```
ldapsearch -L -h ldaphost -p ldapport -b cn=monitor -s one objectclass=*
```

```
dn: cn=backendLDBM-002,cn=monitor
namingcontext: C=AU
namingcontext: C=LDBM
...
searchreferencessent: 0
filter_cache_cache_size: 5000
filter_cache_cache_current: 0
filter_cache_cachehit: 0
filter_cache_cachemiss: 0
filter_cache_cache_percent_hit: 0.00%
filter_cache_cache_refresh: 16487
filter_cache_cache_refresh_avgsize: 0
filter_cache_cache_bypass_limit: 100

dn: cn=backendMonitor,cn=monitor
namingcontext: CN=MONITOR
...

dn: cn=backendSchema,cn=monitor
namingcontext: CN=SCHEMA
...
```

```
searchreferencessent: 0
dn_cache_size: 1000
dn_cache_current: 1000
dn_cachehit: 123743
dn_cachemiss: 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
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
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
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
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
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
currenttimestamp: 20080313132836.785259Z
resettimestamp: 20080313132615.369362Z
resets: 0
numtrimmed: 0
entries: 6
cachesize: 1000
```

Large access groups considerations

Users with large access groups in z/VM LDAP may experience performance problems and increased storage usage in the LDAP server as access groups grow in size.

Some scenarios that require substantial amounts of processing and storage within the z/VM LDAP server, are:

- A search operation which returns all the members of a large access group. This includes either a search which returns the many values with the **member** or **uniqueMember** attribute, or a search which returns the many values in the **ibm-allMembers** operational attribute.
- A search operation which requests all the members of a large access group, but the members are not returned because ACL read permissions prevent the requester from seeing the data.
- Update requests which touch a large access group entry when **persistentSearch 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 z/VM LDAP server may become a factor when there are hundreds of thousands or millions of members in a single access group.

In this case, consider the following corrective actions:

- Increase the LDAP server's virtual storage size, if possible.
- Limit the number of members placed within a single access group and partition the users into separate access groups. The number of members for each access group which can be managed successfully depends on many factors, such as the size of the member values, the amount of virtual storage defined for the z/VM LDAP server, and the level of concurrent activity within the server.
- If possible, avoid configuring **persistentSearch on** for an LDBM backend which 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.

LE heap pools 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 z/VM 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 z/VM LDAP server must process a large access group entry in storage, the following may occur:

- While the request is processing, the z/VM 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.
- Due to the sudden, large demand for storage to process the large group, most or all of the storage available to the z/VM LDAP server may be allocated and reserved to specific heap pools. Although the z/VM LDAP server may appear to be available and able to process a variety of requests, many subsequent requests may fail due to insufficient storage, particularly those for entries with large or numerous attributes. In the absence of any failures, this large increase in storage use by the z/VM LDAP server may be detectable by system resource monitoring products, such as the VM Performance Toolkit.

If these problems occur, consider either tuning the heap pool sizes or disabling the heap pools for the z/VM LDAP server.

Tuning the heap pool sizes optimizes storage usage for the data within the LDAP server. See *z/OS Language Environment Programming Guide* for details on how to tune the 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 which cause the large demands for storage. Note that it is recommended that the storage reports needed for the tuning procedure be gathered in a non-production environment because tracking the storage statistics significantly impacts performance.

Disabling 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 run-time option ’HEAPPOOLS’. This option can be specified on the :parms. tag in the DTCPARMS file. For more details on setting this parameter, see *z/OS: Language Environment Programming Reference* and *z/VM: Language Environment User’s Guide*.

GDBM (Changelog) performance considerations

The GDBM database is used only for the changelog function. By its very nature, this function tends to have a high intensity of update activity compared to read activity. Since update activity is generally more costly than read activity, this function should only be enabled when its use is actually needed.

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 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 which 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 for more information.

Appendix A. Initial LDAP server schema

This appendix shows the initial schema established when the LDAP server is first started. The initial schema is always part of the LDAP server schema and the elements in the initial schema cannot be deleted. With several exceptions, the initial schema cannot be modified. See Updating the schema for more information.

```
cn=schema
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=( 1.3.6.1.4.1.1466.115.121.1.3 DESC 'Attribute type description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.5 DESC 'Binary' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.7 DESC 'Boolean' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.12 DESC 'Distinguished name' )
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=( 1.3.6.1.4.1.1466.115.121.1.24 DESC 'Generalized time' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.26 DESC 'IA5 string' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.27 DESC 'Integer' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.30 DESC 'Matching rule description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.31 DESC 'Matching rule use description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.35 DESC 'Name form description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.37 DESC 'Object class description' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.38 DESC 'Object identifier' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.40 DESC 'Octet string' )
ldapsyntaxes=( 1.3.6.1.4.1.1466.115.121.1.50 DESC 'Telephone 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' )
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=( 1.3.18.0.2.4.405 NAME ( 'distinguishedNameOrderingMatch' )
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
matchingrules=( 1.3.18.0.2.22.2 NAME ( 'ibm-entryUuidMatch' ) SYNTAX 1.3.18.0.2.8.3 )
matchingrules=( 2.5.13.0 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=( 2.5.13.2 NAME ( 'caseIgnoreMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
matchingrules=( 2.5.13.3 NAME ( 'caseIgnoreOrderingMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
matchingrules=( 2.5.13.4 NAME ( 'caseIgnoreSubstringsMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
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=( 2.5.13.13 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=( 2.5.13.17 NAME ( 'octetStringMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 )
matchingrules=( 2.5.13.20 NAME ( 'telephoneNumberMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.50 )
matchingrules=( 2.5.13.21 NAME ( 'telephoneNumberSubstringsMatch' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.50 )
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=( 2.5.13.30 NAME ( 'objectIdentifierFirstComponentMatch' )
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )
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=( 1.2.840.113556.1.4.77 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=( 1.2.840.113556.1.4.656
 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=( 1.2.840.113556.1.4.867 NAME ( 'altSecurityIdentities' ) DESC 'Alternate security identities'
 EQUALITY caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 USAGE userApplications )
attributetypes=( 1.3.6.1.1.4 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=( 1.3.6.1.4.1.1466.101.120.5 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=( 1.3.6.1.4.1.1466.101.120.13 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=( 1.3.18.0.2.4.155 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=( 1.3.18.0.2.4.185 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=( 1.3.18.0.2.4.192 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=( 1.3.18.0.2.4.201 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=( 1.3.18.0.2.4.224 NAME ( 'SAFHoldClass' ) SINGLE-VALUE
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.225 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=( 1.3.18.0.2.4.233 NAME ( 'SAFTsoSecurityLabel' ) SINGLE-VALUE
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.234 NAME ( 'racfPrimaryLanguage' ) SINGLE-VALUE
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.235 NAME ( 'racfSecondaryLanguage' ) DESC 'Secondary language'
  SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.236 NAME ( 'racfOperatorIdentification' ) SINGLE-VALUE
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.237 NAME ( 'racfOperatorClass' )
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
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=( 1.3.18.0.2.4.239 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=( 1.3.18.0.2.4.241 NAME ( 'racfStorageKeyword' ) SINGLE-VALUE
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.242 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=( 1.3.18.0.2.4.244 NAME ( 'racfLevelKeyword' )
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
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=( 1.3.18.0.2.4.249 NAME ( 'racfDOMKeyword' ) SINGLE-VALUE
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.250 NAME ( 'racfKEYKeyword' ) SINGLE-VALUE
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.251 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=( 1.3.18.0.2.4.302 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=( 1.3.18.0.2.4.304 NAME ( 'replicaUpdateTimeInterval' )
  DESC 'Specifies replication update interval' SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  USAGE directoryOperation )
attributetypes=( 1.3.18.0.2.4.470 NAME ( 'ibmAttributeTypes' )
  DESC 'IBM attribute types' SYNTAX 1.3.18.0.2.8.1 USAGE directoryOperation )
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=( 1.3.18.0.2.4.1068 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=( 1.3.18.0.2.4.1780 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=( 1.3.18.0.2.4.1913 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=( 1.3.18.0.2.4.2007 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=( 1.3.18.0.2.4.2239 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=( 1.3.18.0.2.4.2240 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=( 1.3.18.0.2.4.2449 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=( 1.3.18.0.2.4.2481 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=( 1.3.18.0.2.4.3081 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.'
  EQUALITY caseExactMatch SINGLE-VALUE SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  USAGE userApplications )
attributetypes=( 1.3.18.0.2.4.3152 NAME ( 'ibm-slapdReplMaxErrors' )
  DESC 'Limit to allowed errors per replication agreement, 0=unlimited. 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=( 1.3.18.0.2.4.3239 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=( 1.3.18.0.2.4.3342 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=( 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=( 2.5.4.3 NAME ( 'cn' 'commonName' ) SUP name 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 )
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=( 2.5.4.31 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=( 2.5.18.1 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=( 2.5.18.2 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=( 2.5.18.10 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=( 2.5.21.1 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=( 2.16.840.1.113730.3.1.5 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=( 2.16.840.1.113730.3.1.8 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=( 2.16.840.1.113730.3.1.10 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=( 2.16.840.1.113730.3.1.34 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=( 2.16.840.1.113730.3.1.77 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=( 2.16.840.1.113730.3.1.198 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 )
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=( 1.2.840.113556.1.4.77 ACCESS-CLASS normal )
ibmattributetypes=( 1.2.840.113556.1.4.656 ACCESS-CLASS normal )
ibmattributetypes=( 1.2.840.113556.1.4.867 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.1.4 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.1.5 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.6.1.4.1.1466.101.120.5 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=( 1.3.6.1.4.1.1466.101.120.13 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=( 1.3.18.0.2.4.155 ACCESS-CLASS critical )
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=( 1.3.18.0.2.4.192 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=( 1.3.18.0.2.4.201 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=( 1.3.18.0.2.4.224 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.225 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.226 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.227 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.228 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.229 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.230 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.231 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.232 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.233 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.234 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.235 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.236 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.237 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.238 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.239 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.240 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.241 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.242 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.243 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.244 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.245 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.246 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.247 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.248 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.249 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.250 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.251 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.252 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.253 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.254 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.255 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.256 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=( 1.3.18.0.2.4.302 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.303 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.304 ACCESS-CLASS normal )
ibmattributetypes=( 1.3.18.0.2.4.470 ACCESS-CLASS system )
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=( 1.3.18.0.2.4.1068 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=( 1.3.18.0.2.4.1780 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.1913 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2007 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2239 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2240 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2241 ACCESS-CLASS sensitive )
ibmattributetypes=( 1.3.18.0.2.4.2242 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=( 1.3.18.0.2.4.2449 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2481 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.2482 ACCESS-CLASS system )
ibmattributetypes=( 1.3.18.0.2.4.3081 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=( 1.3.18.0.2.4.3152 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=( 1.3.18.0.2.4.3239 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=( 1.3.18.0.2.4.3342 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=( 2.5.4.0 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.1 ACCESS-CLASS normal )
ibmattributetypes=( 2.5.4.3 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 )
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=( 2.5.4.31 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=( 2.5.18.1 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=( 2.5.18.10 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=( 2.16.840.1.113730.3.1.5 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=( 2.16.840.1.113730.3.1.8 ACCESS-CLASS sensitive )
ibmattributetypes=( 2.16.840.1.113730.3.1.9 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=( 2.16.840.1.113730.3.1.34 ACCESS-CLASS normal )
ibmattributetypes=( 2.16.840.1.113730.3.1.35 ACCESS-CLASS normal )
ibmattributetypes=( 2.16.840.1.113730.3.1.77 ACCESS-CLASS normal )
ibmattributetypes=( 2.16.840.1.113730.3.1.198 ACCESS-CLASS normal )
objectclasses=( 1.3.18.0.2.6.28  NAME ( 'container' ) DESC 'An object that can contain other objects'
  STRUCTURAL SUP ( top ) MUST ( cn ) )
objectclasses=( 1.3.18.0.2.6.55  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=( 1.3.18.0.2.6.56  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=( 1.3.18.0.2.6.58  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=( 1.3.18.0.2.6.59  NAME ( 'racfGroup' ) DESC 'Represents a RACF GROUP Profile entry'
  STRUCTURAL SUP ( racfBaseCommon ) MUST ( racfid ) MAY ( racfSuperiorGroup $ racfGroupNoTermUAC $
  racfSubGroupName $ racfGroupUserids $ racfGroupUniversal ) )
objectclasses=( 1.3.18.0.2.6.60  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=( 1.3.18.0.2.6.64  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=( 1.3.18.0.2.6.66  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=( 1.3.18.0.2.6.67  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 $ seeAlso $
  replicaPort $ replicaBindMethod $ replicaUseSSL $ replicaUpdateTimeInterval ) )
objectclasses=( 1.3.18.0.2.6.74  NAME ( 'aliasObject' ) DESC 'Defines an alias for a directory entry'
  AUXILIARY SUP ( top ) MUST ( aliasedObjectName ) )
objectclasses=( 1.3.18.0.2.6.75  NAME ( 'accessGroup' ) DESC 'Group used for access control' STRUCTURAL
  SUP ( top ) MUST ( cn ) MAY ( member $ businessCategory $ seeAlso $ owner $ ou $ o $ description ) )
objectclasses=( 1.3.18.0.2.6.76  NAME ( 'accessRole' ) DESC 'Role used for access control' STRUCTURAL
  SUP ( top ) MUST ( cn ) MAY ( member $ businessCategory $ seeAlso $ owner $ ou $ o $ description ) )
objectclasses=( 1.3.18.0.2.6.174  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=( 1.3.18.0.2.6.447  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=( 1.3.6.1.4.1.1466.101.120.111  NAME ( 'extensibleObject' )
  DESC 'Permits the entry to hold any attribute type defined in the schema' AUXILIARY SUP ( top ) )
objectclasses=( 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=( 2.5.6.9  NAME ( 'groupOfNames' ) DESC 'Defines entries for a group of names' STRUCTURAL
  SUP ( top ) MUST ( cn $ member ) MAY ( businessCategory $ seeAlso $ owner $ ou $ o $ description ) )
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=( 2.5.17.0  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=( 2.16.840.1.113730.3.2.1  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=( 2.16.840.1.113730.3.2.6  NAME ( 'referral' )
  DESC 'Defines a pointer to another server' STRUCTURAL SUP ( top ) MUST ( ref ) )
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 ) )
```

Appendix B. Supported server controls

The sections that follow describe the supported server controls. For information on ASN.1 (Abstract Syntax Notation One) and BER (Basic Encoding Rules), go to the following Web site:

`ftp://ftp.rsa.com/pub/pkcs/ascii/layman.asc`

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.

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 may be embedded in DN-syntax attributes throughout the directory contents. It may be desirable to replace the embedded values with their renamed counterparts (realignment). The presence of this control on the Modify DN request causes the server to realign matching attribute values in all attribute types whose syntax is **Distinguished Name** (OID 1.3.6.1.4.1.1466.115.121.1.12), as well as in the attribute types of **aclEntry** and **entryOwner**, which are known to contain distinguished names. The server will evaluate whether the bound user has permission to modify the candidate attribute values, as determined by the appropriate access controls and the permissions granted by those access controls to the bound DN. If the permissions granted to the bound DN are sufficient to modify the candidate attribute values, those values will be realigned to match their respective new DN values. If any single access check fails, the entire 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 will not be 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 may be long-running operations if they affect many entries in the directory (for example, if they rename an entry with a subtree containing many subordinate entries), so it may be desirable to limit the duration of the operation. The presence of this control on the Modify DN request causes the operation to be abandoned by the server if the number of seconds specified in the control value is exceeded. When the operation is abandoned, all changes to the directory associated with the Modify DN operation are undone. 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 for more information on 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 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.

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, 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 on the **persistentSearch** configuration option.

• **Server behavior:** The server behaves as described in the specification found at http://www.mozilla.org/directory/ietf-docs/draft-smith-psearch-ldap-01.txt, with the following exceptions:

1. An error is returned if an error occurs during processing of the persistent search request. Section 4.b of the specification indicates that SearchResultsDone message is not returned if a persistent search is requested. This is not recognized in the case of an error.
2. If more than one **PersistentSearchControl** is received per search request, LDAP_PROTOCOL_ERROR is returned.
3. If the requesting client is not bound as adminDN, LDAP_UNWILLING_TO_PERFORM is returned.
4. If persistent search is requested and the 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 afterwards. No search results are returned for entries moved out of the search filter or scope due to 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 normal search will be performed for those backends.
9. If a **PersistentSearch** control is included in a search request for an LDBM, or GDBM backend that has not enabled persistent search, the search request is rejected with LDAP_UNAVAILABLE_CRITICAL_EXTENTION (0x35) if the control is critical. If the control is not critical, a 'normal' search is performed (even if changesOnly is **TRUE**).
10. Change log entries trimmed by the LDAP server due to the **changeLogMaxAge** or **changeLogMaxEntries** configuration options are not returned to a persistent search of the change log directory.
11. If the **manageDsaIT** control is not specified with the **PersistentSearch** control and phase one of the search finds a referral, the referral is returned to the client. If the base of the search is equal to or below a referral, the referral is returned and the persistent search second phase does not occur. During the second phase of persistent search, referral entries are always processed like normal entries, even if the **manageDsaIT** control is not specified on the persistent search.
12. Idle connection time out also affects persistent search connections. See the description of the **idleConnectionTimeout** configuration option in "Step 7. 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 are respected only during the first phase of persistent search, when existing entries are searched. An error is returned if either limit is exceeded and the persistent search ends. During the second phase, when changed entries are searched, **sizeLimit** and **timeLimit** are ignored.
14. Only the entry specified in a 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, or user-group connection, 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, or connection profile.
16. Operational attributes are returned on persistent searches except the following: **aclSource, ownerSource, ibm-allGroups**, and **ibm-allMembers**. 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.

replicateOperationalAttributes

- **Name: replicateOperationalAttributes**

- **Description:** Used to pass the values of operational attributes that are normally 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 normally 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 so that the entry on the replica will have the same values as on the master.
- **Server behavior:**
  1. The control is only supported on an add, modify, or modify DN request on a peer or read-only replica server. If the control is specified on another request and the control is critical, the server returns LDAP_UNAVAILABLE_CRITICAL_EXTENSION.
  2. The requestor 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** option in the backend section of the LDAP server configuration file. If the requestor is not bound in this way 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.

Appendix C. Related Protocol Specifications

IBM is committed to industry standards. The internet protocol suite is still evolving through Requests for Comments (RFC). New protocols are being designed and implemented by researchers, and are brought to the attention of the internet community in the form of RFCs. Some of these are so useful that they become a recommended protocol. That is, all future implementations for TCP/IP are recommended to implement this particular function or protocol. These become the *de facto* standards, on which the TCP/IP protocol suite is built.

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
1013	*X Window System Protocol, Version 11: Alpha Update*	R.W. Scheifler
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

RFCs

RFC	Title	Author
1034	*Domain Names—Concepts and Facilities*	P.V. Mockapetris
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
1044	*Internet Protocol on Network System's HYPERchannel: Protocol Specification*	K. Hardwick, J. Lekashman
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
1198	*FYI on the X Window System*	R.W. Scheifler
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
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

RFC	Title	Author
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
1374	*IP and ARP on HIPPI*	J. Renwick, A. Nicholson
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
2060	*IMAP Version 4 Protocol Specification*	M. Crispin
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
2104	*HMAC: Keyed-Hashing for Message Authentication*	H. Krawczyk, M. Bellare, R. Canetti
2195	*IMAP/POP AUTHorize Extension for Simple Challenge/Response*	J. Klensin, R. Catoe, P. Krumviede
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

RFCs

RFC	Title	Author
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(tm) 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
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

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 through http://www.ietf.org/.

Appendix D. Abbreviations and Acronyms

The following abbreviations and acronyms are used throughout this book.

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
CLAW	Common Link Access to Workstation
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
FTAM	File Transfer Access Management
FTP	File Transfer Protocol
FTP API	File Transfer Protocol Applications Programming Interface
GCS	Group Control System
GDDM®	Graphical Data Display Manager
GDDMXD	Graphics Data Display Manager Interface for X Window System
GDF	Graphics Data File
HCH	HYPERchannel device
HIPPI	High Performance Parallel Interface
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
LCS	IBM LAN Channel Station
LDAP	Lightweight Directory Access Protocol
LPD	Line Printer Daemon
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
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®
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 U.S.A. IBM may not offer the products, services, or features discussed in this document in all 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
Armonk, New York 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs

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

Glossary

For a list of z/VM terms and their definitions, see *z/VM: Glossary*.

The z/VM glossary is also available through the online z/VM HELP Facility. For example, to display the definition of the term “dedicated device”, issue the following HELP command:

```
help glossary dedicated device
```

While you are in the glossary help file, you can do additional searches:

- To display the definition of a new term, type a new HELP command on the command line:

  ```
  help glossary newterm
  ```

  This command opens a new help file inside the previous help file. You can repeat this process many times. The status area in the lower right corner of the screen shows how many help files you have open. To close the current file, press the Quit key (PF3/F3). To exit from the HELP Facility, press the Return key (PF4/F4).
- To search for a word, phrase, or character string, type it on the command line and press the Clocate key (PF5/F5). To find other occurrences, press the key multiple times.

  The Clocate function searches from the current location to the end of the file. It does not wrap. To search the whole file, press the Top key (PF2/F2) to go to the top of the file before using Clocate.

Bibliography

See the following publications for additional information about z/VM. For abstracts of the z/VM publications, see *z/VM: General Information*.

Where to Get z/VM Information

z/VM product information is available from the following sources:

- z/VM Information Center at publib.boulder.ibm.com/infocenter/zvm/v6r1/index.jsp
- z/VM Internet Library at www.ibm.com/eserver/zseries/zvm/library/
- IBM Publications Center at www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss
- *IBM Online Library: z/VM Collection on DVD*, SK5T-7054

z/VM Base Library

Overview

- *z/VM: General Information*, GC24-6193
- *z/VM: Glossary*, GC24-6195
- *z/VM: License Information*, GC24-6200

Installation, Migration, and Service

- *z/VM: Guide for Automated Installation and Service*, GC24-6197
- *z/VM: Migration Guide*, GC24-6201
- *z/VM: Service Guide*, GC24-6232
- *z/VM: VMSES/E Introduction and Reference*, GC24-6243

Planning and Administration

- *z/VM: CMS File Pool Planning, Administration, and Operation*, SC24-6167
- *z/VM: CMS Planning and Administration*, SC24-6171
- *z/VM: Connectivity*, SC24-6174
- *z/VM: CP Planning and Administration*, SC24-6178
- *z/VM: Getting Started with Linux on System z*, SC24-6194
- *z/VM: Group Control System*, SC24-6196
- *z/VM: I/O Configuration*, SC24-6198
- *z/VM: Running Guest Operating Systems*, SC24-6228
- *z/VM: Saved Segments Planning and Administration*, SC24-6229
- *z/VM: Secure Configuration Guide*, SC24-6230
- *z/VM: TCP/IP LDAP Administration Guide*, SC24-6236
- *z/VM: TCP/IP Planning and Customization*, SC24-6238
- *z/OS and z/VM: Hardware Configuration Manager User's Guide*, SC33-7989

Customization and Tuning

- *z/VM: CP Exit Customization*, SC24-6176
- *z/VM: Performance*, SC24-6208

Operation and Use

- *z/VM: CMS Commands and Utilities Reference*, SC24-6166
- *z/VM: CMS Pipelines Reference*, SC24-6169
- *z/VM: CMS Pipelines User's Guide*, SC24-6170
- *z/VM: CMS Primer*, SC24-6172
- *z/VM: CMS User's Guide*, SC24-6173
- *z/VM: CP Commands and Utilities Reference*, SC24-6175
- *z/VM: System Operation*, SC24-6233
- *z/VM: TCP/IP User's Guide*, SC24-6240
- *z/VM: Virtual Machine Operation*, SC24-6241
- *z/VM: XEDIT Commands and Macros Reference*, SC24-6244
- *z/VM: XEDIT User's Guide*, SC24-6245
- *CMS/TSO Pipelines: Author's Edition*, SL26-0018

Application Programming

- *z/VM: CMS Application Development Guide*, SC24-6162
- *z/VM: CMS Application Development Guide for Assembler*, SC24-6163
- *z/VM: CMS Application Multitasking*, SC24-6164
- *z/VM: CMS Callable Services Reference*, SC24-6165
- *z/VM: CMS Macros and Functions Reference*, SC24-6168

- *z/VM: CP Programming Services*, SC24-6179
- *z/VM: CPI Communications User's Guide*, SC24-6180
- *z/VM: Enterprise Systems Architecture/ Extended Configuration Principles of Operation*, SC24-6192
- *z/VM: Language Environment User's Guide*, SC24-6199
- *z/VM: OpenExtensions Advanced Application Programming Tools*, SC24-6202
- *z/VM: OpenExtensions Callable Services Reference*, SC24-6203
- *z/VM: OpenExtensions Commands Reference*, SC24-6204
- *z/VM: OpenExtensions POSIX Conformance Document*, GC24-6205
- *z/VM: OpenExtensions User's Guide*, SC24-6206
- *z/VM: Program Management Binder for CMS*, SC24-6211
- *z/VM: Reusable Server Kernel Programmer's Guide and Reference*, SC24-6220
- *z/VM: REXX/VM Reference*, SC24-6221
- *z/VM: REXX/VM User's Guide*, SC24-6222
- *z/VM: Systems Management Application Programming*, SC24-6234
- *z/VM: TCP/IP Programmer's Reference*, SC24-6239
- *Common Programming Interface Communications Reference*, SC26-4399
- *Common Programming Interface Resource Recovery Reference*, SC31-6821
- *z/OS: IBM Tivoli Directory Server Plug-in Reference for z/OS*, SA76-0148
- *z/OS: Language Environment Concepts Guide*, SA22-7567
- *z/OS: Language Environment Debugging Guide*, GA22-7560
- *z/OS: Language Environment Programming Guide*, SA22-7561
- *z/OS: Language Environment Programming Reference*, SA22-7562
- *z/OS: Language Environment Run-Time Messages*, SA22-7566
- *z/OS: Language Environment Writing ILC Applications*, SA22-7563
- *z/OS MVS Program Management: Advanced Facilities*, SA22-7644
- *z/OS MVS Program Management: User's Guide and Reference*, SA22-7643

Diagnosis

- *z/VM: CMS and REXX/VM Messages and Codes*, GC24-6161
- *z/VM: CP Messages and Codes*, GC24-6177
- *z/VM: Diagnosis Guide*, GC24-6187
- *z/VM: Dump Viewing Facility*, GC24-6191
- *z/VM: Other Components Messages and Codes*, GC24-6207
- *z/VM: TCP/IP Diagnosis Guide*, GC24-6235
- *z/VM: TCP/IP Messages and Codes*, GC24-6237
- *z/VM: VM Dump Tool*, GC24-6242
- *z/OS and z/VM: Hardware Configuration Definition Messages*, SC33-7986

z/VM Facilities and Features

Data Facility Storage Management Subsystem for VM

- *z/VM: DFSMS/VM Customization*, SC24-6181
- *z/VM: DFSMS/VM Diagnosis Guide*, GC24-6182
- *z/VM: DFSMS/VM Messages and Codes*, GC24-6183
- *z/VM: DFSMS/VM Planning Guide*, SC24-6184
- *z/VM: DFSMS/VM Removable Media Services*, SC24-6185
- *z/VM: DFSMS/VM Storage Administration*, SC24-6186

Directory Maintenance Facility for z/VM

- *z/VM: Directory Maintenance Facility Commands Reference*, SC24-6188
- *z/VM: Directory Maintenance Facility Messages*, GC24-6189
- *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*, SC24-6190

Open Systems Adapter/Support Facility

- *System z10, System z9 and eServer zSeries: Open Systems Adapter-Express Customer's Guide and Reference*, SA22-7935
- *System z9 and eServer zSeries 890 and 990: Open Systems Adapter-Express Integrated Console Controller User's Guide*, SA22-7990

- *System z: Open Systems Adapter-Express Integrated Console Controller 3215 Support*, SA23-2247

Performance Toolkit for VM™

- *z/VM: Performance Toolkit Guide*, SC24-6209
- *z/VM: Performance Toolkit Reference*, SC24-6210

RACF® Security Server for z/VM

- *z/VM: RACF Security Server Auditor's Guide*, SC24-6212
- *z/VM: RACF Security Server Command Language Reference*, SC24-6213
- *z/VM: RACF Security Server Diagnosis Guide*, GC24-6214
- *z/VM: RACF Security Server General User's Guide*, SC24-6215
- *z/VM: RACF Security Server Macros and Interfaces*, SC24-6216
- *z/VM: RACF Security Server Messages and Codes*, GC24-6217
- *z/VM: RACF Security Server Security Administrator's Guide*, SC24-6218
- *z/VM: RACF Security Server System Programmer's Guide*, SC24-6219
- *z/VM: Security Server RACROUTE Macro Reference*, SC24-6231

Remote Spooling Communications Subsystem Networking for z/VM

- *z/VM: RSCS Networking Diagnosis*, GC24-6223
- *z/VM: RSCS Networking Exit Customization*, SC24-6224
- *z/VM: RSCS Networking Messages and Codes*, GC24-6225
- *z/VM: RSCS Networking Operation and Use*, SC24-6226
- *z/VM: RSCS Networking Planning and Configuration*, SC24-6227
- *Network Job Entry: Formats and Protocols*, SA22-7539

Prerequisite Products

Device Support Facilities

- *Device Support Facilities: User's Guide and Reference*, GC35-0033

Environmental Record Editing and Printing Program

- *Environmental Record Editing and Printing Program (EREP): Reference*, GC35-0152
- *Environmental Record Editing and Printing Program (EREP): User's Guide*, GC35-0151

Other TCP/IP Related Publications

This section lists other publications, outside the z/VM V6.1 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

Program Number: 5741-A07

Printed in USA

SC24-6236-00