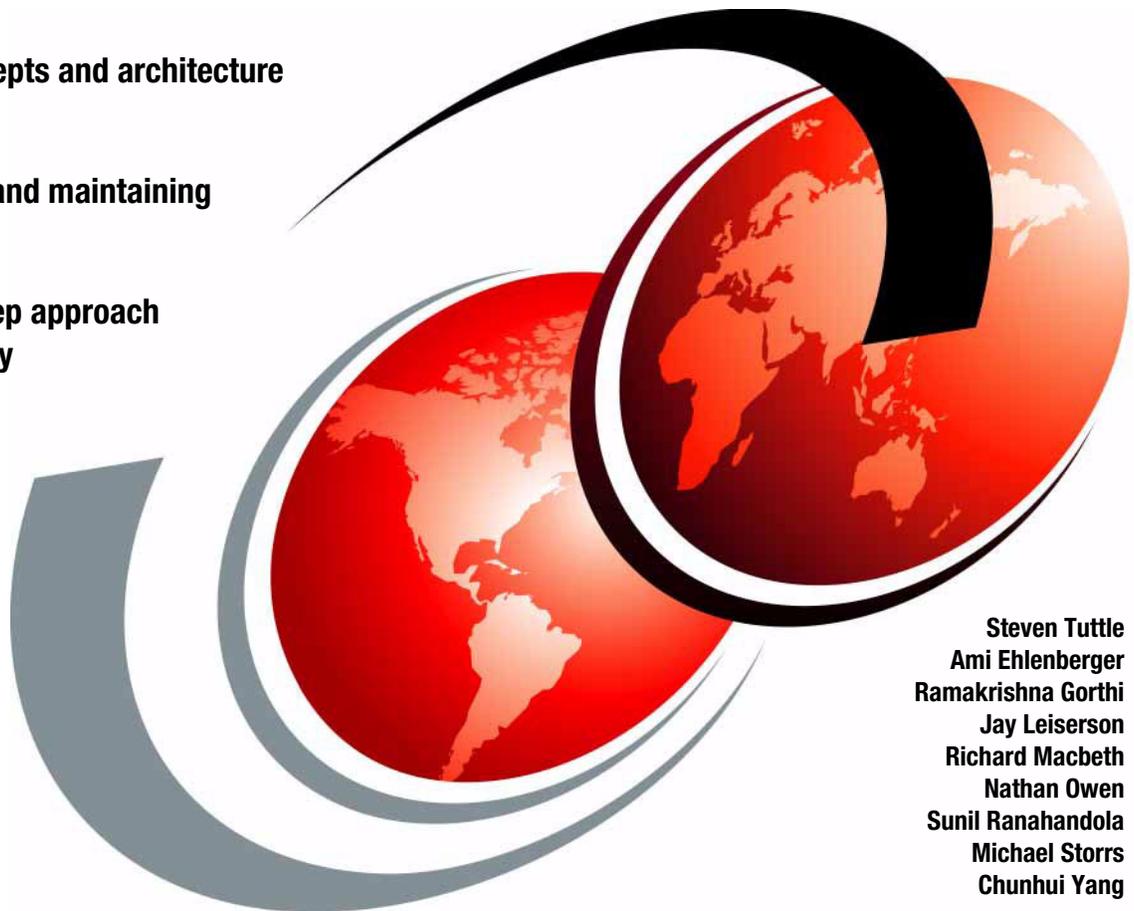IBM

Understanding LDAP
Design and Implementation

LDAP concepts and architecture

**Designing and maintaining
LDAP**

**Step-by-step approach
for directory**

Steven Tuttle
Ami Ehlenberger
Ramakrishna Gorthi
Jay Leiserson
Richard Macbeth
Nathan Owen
Sunil Ranahandola
Michael Storrs
Chunhui Yang

Redbooks

IBM

International Technical Support Organization

Understanding LDAP Design and Implementation

June 2004

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

Second Edition (June 2004)

This edition applies to Version 5, Release 2 of IBM Tivoli Directory Server.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

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.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®	MVS™	SecureWay®
Cloudscape™	Notes®	SP2®
DB2 Universal Database™	OS/390®	Tivoli Enterprise™
DB2®	OS/400®	Tivoli®
Domino®	pSeries®	WebSphere®
IBM®	RACF®	World Registry™
ibm.com®	RDN™	xSeries®
iSeries™	Redbooks (logo) ™	z/OS®
Lotus Notes®	Redbooks™	zSeries®
Lotus®	Sametime®	

The following terms are trademarks of other companies:

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Preface

Lightweight Directory Access Protocol (LDAP) is a fast growing technology for accessing common directory information. LDAP has been embraced and implemented in most network-oriented middleware. As an open, vendor-neutral standard, LDAP provides an extendable architecture for centralized storage and management of information that needs to be available for today's distributed systems and services.

After a fast start, it can be assumed that LDAP has become the de facto access method for directory information, much the same as the Domain Name System (DNS) is used for IP address look-up on almost any system on an intranet and on the Internet. LDAP is currently supported in most network operating systems, groupware and even shrink-wrapped network applications.

This book was written for those readers who need to understand the basic principles and concepts of LDAP. Some background knowledge about heterogeneous, distributed systems is assumed and highly beneficial when reading this book. This book is not meant to be an LDAP implementation guide, nor does it contain product-related or vendor-specific information other than as used in examples.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Steven Tuttle is a Project Leader for the International Technical Support Organization (ITSO), Austin Center. He has 13 years of experience in the IT industry. He has worked at IBM® for 10 years, with five years of experience with IBM security products. He holds a degree in Computer Science from Clarkson University in Potsdam, New York, with concentrations in Mathematics and Psychology. His areas of expertise include the IBM Tivoli® Enterprise™ products and the IBM Tivoli Security products. Before joining the ITSO, he worked for IBM Tivoli Services in the Security Practice as an enterprise security solution designer using IBM Tivoli software products.

Ami Ehlenberger has been with IBM for the past five years. Her career has included working in OS/390® development, z/OS® Integration Test, and the zSeries® Custom Technology Center. Her technical concentration is Internet security, designing solutions that focus on WebSphere®, LDAP, and Tivoli

security products. Ami has a BS in Computer Science from Indiana University of Pennsylvania and an MBA in e-Business from the University of Phoenix. Ami currently manages the IBM Server and Technology Group's zSeries Services Team. The team specializes in Web enablement and solution design, concentrating on the zSeries platform.

Ramakrishna Gorthi is a developer for the IBM Tivoli Directory Server, Pune Center in India. He has worked at IBM for two and a half years, with one year of Level 2 Customer Support for the various versions of the IBM Tivoli Directory Server. He holds a degree in Computer Engineering from Pune Institute of Computer Technology, Pune (India). His areas of expertise include the IBM Tivoli Directory Server from the Tivoli Security Products. Apart from the immense experience gained as a Customer Support Representative, he has also earned a good reputation in the different phases of the product life cycle for the IBM Tivoli Directory Server, like development and testing.

Jay Leiserson is a Solution Architect for Tivoli Security products. He has twenty-five years of experience in systems analysis, solution design, and software development. He has worked at IBM for 24 years and has an extensive and varied background that includes directory design and integration, identity management solution design, Internet security, and application and operating system development for distributed systems. He holds a degree in Economics from Antioch College in Yellow Springs, Ohio.

Richard Macbeth is an IBM Directory Services Architect for Tivoli Services, Americas Security Practice. He has been with IBM for 25 years in the computer/IT field with 12 years of experience in the LDAP Directory field. He has current certifications with Novell as a Certified Directory Engineer, Certified Novell Instructor, Certified Novell Engineer, and Sun One Directory 5 Engineer. He has worked on a number of versions of SecureWay®/IBM Directory Server on most platforms. He also has four years of experience with Tivoli Access Manager and one year of experience with IBM Directory Integrator. He also held a CCNP Certification with Cisco and had over 10 years of experience as a Senior Network IT Specialist.

Nathan Owen is a Identity Management Architect within IBM Software Group. Nathan has worked in the Identity Management space for over eight years with a particular focus on directory service related technologies such as X.500/LDAP directories, Meta-directories, and Virtual Directories. He took a three year pause from IBM in 1999 and co-founded virtual directory vendor Octet String Inc., before returning to IBM late in 2002. He holds Political Science degree from Central Michigan University in Mt. Pleasant, Michigan. His areas of expertise include IBM Tivoli Directory Server (ITDS), IBM Tivoli Directory Integrator (ITDI), as well as the other the products in the Tivoli Identity Management portfolio.

Sunil Ranahandola is a Software Engineer for the IBM Global Services (IGSI), India Center. He started his career with IBM in March 2001 and has been working with IBM since then. He has almost three years of experience in the IT industry. He holds a degree in Computer Science from University College of Engineering, Burla, Orissa, India. His areas of expertise include the IBM Tivoli Directory Services.

Michael Storrs is an IT Specialist for the Tivoli Security Group. He has seven years of experience in the IT industry, and has worked with enterprise access and identity management products for the last five years. He holds a degree in Electrical Engineering from the University of Virginia. His areas of expertise include the Tivoli Security Products, IBM Tivoli Directory Integrator, directory servers, and application development.

Chunhui Yang is a Metadata Architect and Directory Consultant in IBM Software Group, RTP. She has direct experience with the full project lifecycle of information systems for Microsoft®, Dow Jones, Reuters, and IBM, and is recognized as a chief contributor with National awards to many projects in areas of system architecture design, development and deployment on Directory solutions and n-tier Web-based application solutions.

Thanks to the following people for their contributions to this project:

Tony Bhe, Tamikia Barrow, Linda Robinson, Margaret Ticknor
International Technical Support Organization, Austin Center

Julie Czubik
International Technical Support Organization, Poughkeepsie Center

Chris Ehrsam
IBM Directory Solutions Architect

John McGarvey
IBM Directory Solutions Architect/Security Integration

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

      ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

► Use the online **Contact us** review redbook form found at:

      ibm.com/redbooks

► Send your comments in an Internet note to:

      redbook@us.ibm.com

► Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. JN9B Building 003 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition may also include minor corrections and editorial changes that are not identified.

Summary of Changes
for SG24-4986-01
for Understanding LDAP
as created or updated on July 18, 2006.

June 2004, Second Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

New information
- ▶ IBM Tivoli Directory Integrator information
- ▶ Information on zSeries and Intel® Linux

Changed information
- ▶ Updated information to latest release of products

Part 1

Directories and LDAP

In this part we introduce directories and LDAP. Specifically, we provide an introduction to LDAP, cover LDAP concepts and architecture, and provide some information on how to plan for a directory deployment in your environment.

1

Introduction to LDAP

Today people and businesses rely on networked computer systems to support distributed applications. These distributed applications might interact with computers on the same local area network, within a corporate intranet, within extranets linking up partners and suppliers, or anywhere on the worldwide Internet. To improve functionality and ease-of-use, and to enable cost-effective administration of distributed applications, information about the services, resources, users, and other objects accessible from the applications needs to be organized in a clear and consistent manner. Much of this information can be shared among many applications, but it must also be protected in order to prevent unauthorized modification or the disclosure of private information.

Information describing the various users, applications, files, printers, and other resources accessible from a network is often collected into a special database that is sometimes called a directory. As the number of different networks and applications has grown, the number of specialized directories of information has also grown, resulting in islands of information that are difficult to share and manage. If all of this information could be maintained and accessed in a consistent and controlled manner, it would provide a focal point for integrating a distributed environment into a consistent and seamless system.

The Lightweight Directory Access Protocol (LDAP) is an open industry standard that has evolved to meet these needs. LDAP defines a standard method for accessing and updating information in a directory. LDAP has gained wide acceptance as the directory access method of the Internet and is therefore also

becoming strategic within corporate intranets. It is being supported by a growing number of software vendors and is being incorporated into a growing number of applications. For example, the two most popular Web browsers, Netscape Navigator/Communicator and Microsoft Internet Explorer, as well as application middleware, such as the IBM WebSphere Application Server or the IBM HTTP server, support LDAP functionality as a base feature.

This chapter introduces the fundamentals of directories and the most commonly used protocol to access directories, the LDAP protocol. You will also learn about the various components that make up a directory.

Part of the information covered in this chapter and further information on LDAP directory concepts and implementations can be found in the following publications:

► *Implementation and Practical Use of LDAP on the IBM iSeries™* Server, SG24-6193

► *Using LDAP for Directory Integration*, SG24-6163

Another book that contains good information about directory concepts and architecture is *e-Directories Enterprise Software, Solutions*, and Services, ISBN 0-201-70039-5.

1.1 Directories

A directory is a listing of information about objects arranged in some order that gives details about each object. Common examples are a city telephone directory and a library card catalog. For a telephone directory, the objects listed are people; the names are arranged alphabetically, and the details given about each person are address and telephone number. Books in a library card catalog are ordered by author or by title, and information such as the ISBN number of the book and other publication information is given.

In computer terms, a directory is a specialized database, also called a data repository, that stores typed and ordered information about objects. A particular directory might list information about printers (the objects) consisting of typed information such as location (a formatted character string), speed in pages per minute (numeric), print streams supported (for example PostScript or ASCII), and so on.

Directories allow users or applications to find resources that have the characteristics needed for a particular task. For example, a directory of users can be used to look up a person's e-mail address or fax number. A directory could be searched to find a nearby PostScript color printer. Or a directory of application servers could be searched to find a server that can access customer billing information.

The terms *white pages* and *yellow pages* are sometimes used to describe how a directory is used. If the name of an object (person, printer) is known, its characteristics (phone number, pages per minute) can be retrieved. This is similar to looking up a name in the white pages of a telephone directory. If the name of a particular individual object is not known, the directory can be searched for a list of objects that meet a certain requirement. This is like looking up a listing of hairdressers in the yellow pages of a telephone directory. However, directories stored on a computer are much more flexible than the yellow pages of a telephone directory because they can usually be searched by specific criteria, not just by a predefined set of categories.

1.1.1 Directory versus database

A directory is often described as a database, but it is a specialized database that has characteristics that set it apart from general-purpose relational databases. One special characteristic of directories is that they are accessed (read or searched) much more often than they are updated (written). Hundreds of people might look up an individual's phone number, or thousands of print clients might look up the characteristics of a particular printer, but the phone number or printer characteristics rarely change.

Because directories must be able to support high volumes of read requests, they are typically optimized for read access. Write access might be limited to system administrators or to the owner of each piece of information. A general-purpose relational database, on the other hand, needs to support applications, such as airline reservations and banking applications, with relatively high-update volumes.

Because directories are meant to store relatively static information and are optimized for that purpose, they are not appropriate for storing information that changes rapidly. For example, the number of jobs currently in a print queue probably should not be stored in the directory entry for a printer because that information would have to be updated frequently to be accurate. Instead, the directory entry for the printer can contain the network address of a print server. The print server can be queried to get the current queue length if desired. The information in the directory (the print server address) is static, whereas the number of jobs in the print queue is dynamic.

Another difference between directories and general-purpose relational databases is that most directory implementations still do not support transactions. However, transactions are supported in LDAP and are limited to transactions within the LDAP directory, and do not include other transactions (for example, database operations). Transactions are all-or-nothing operations that must be completed in total or not at all. For example, when transferring money from one bank account to another, the money must be debited from one account and credited to the other account in a single transaction. If only half of this transaction completes or someone accesses the accounts while the money is in transit, the accounts will not balance. General-purpose relational databases usually support such transactions, which complicates their implementation.

Because general-purpose relational databases must support arbitrary applications such as banking and inventory control, they allow arbitrary collections of data to be stored. Directories may be limited in the type of data they allow to be stored (although the architecture does not impose such a limitation). For example, a directory specialized for customer contact information might be limited to storing only personal information such as names, addresses, and phone numbers. If a directory is extensible, it can be configured to store a variety of types of information making it more useful to a variety of programs.

Another important difference between a directory and a general-purpose relational database is in the way information can be accessed. Most databases support a standardized, very powerful access method called Structured Query Language (SQL). SQL allows complex update and query functions at the cost of program size and application complexity. Directories, such as an LDAP directory, on the other hand, use a simplified and optimized access protocol that can be used in slim and relatively simple applications.

Because directories are not intended to provide as many functions as general-purpose relational databases, they can be optimized to economically provide more applications with rapid access to directory data in large distributed environments. If your intended use of the directory is to be read, mostly in a non-transactional environment, both the directory client and directory server can be simplified and optimized.

A request is typically performed by the directory client, and the process that looks up information in the directory is called the directory server. In general, servers provide a specific service to clients. Sometimes a server might become the client of other servers in order to gather the information necessary to process a request.

A directory service is only one type of service that might be available in a client/server environment. Other common examples of services are file services, mail services, print services, Web page services, and so on. The client and server processes may or may not be on the same machine. A server is capable of serving many clients. Some servers can process client requests in parallel. Other servers queue incoming client requests for serial processing if they are currently busy processing another client's request.

An API defines the programming interface a particular programming language uses to access a service. The format and contents of the messages exchanged between client and server must adhere to an agreed-upon protocol.

1.1.2 LDAP: Protocol or directory

The Lightweight Directory Access Protocol (LDAP) defines a message protocol used by directory clients and directory servers.The LDAP protocol uses different messages. For example, a bindRequest may be sent from the client to the LDAP server at the beginning of a connection. A searchRequest is used to search for a specific entry in the directory.

There are also associated LDAP APIs for the C language and ways to access LDAP from within a Java™ application. Additionally, within the Microsoft development environment, you can access LDAP directories through its Active Directory Service Interface (ADSI) In general with LDAP, the client is not dependent upon a particular implementation of the server, and the server can implement the directory however it chooses.

LDAP is an open industry standard that defines a standard method for accessing and updating information in a directory. LDAP has gained wide acceptance as the directory access method of the Internet and is therefore also becoming strategic within corporate intranets. It is being supported by a growing number of

software vendors and is being incorporated into a growing number of applications.

LDAP defines a communication protocol. That is, it defines the transport and format of messages used by a client to access data in an X.500-like directory. LDAP does not define the directory service itself. When people talk about the LDAP directory, that is the information that is stored and can be retrieved by the LDAP protocol.

All modern LDAP directory servers are based on LDAP Version 3. You can use a Version 2 client with a Version 3 server. However, you cannot use a Version 3 client with a Version 2 server unless you bind as a Version 2 client and use only Version 2 APIs.

All LDAP servers share many basic characteristics since they are based on the industry standard Request for Comments (RFCs). However, due to implementation differences, they are not all completely compatible with each other when there is not a standard defined.

1.1.3 Directory clients and servers

Directories are usually accessed using the client/server model of communication. An application that wants to read or write information in a directory does not access the directory directly. Instead, it calls a function or application programming interface (API) that causes a message to be sent to another process. This second process accesses the information in the directory on behalf of the requesting application via TCP/IP. The default TCP/IP ports are 636 for secure communications and 389 for unencrypted communications. The results of the read or write action are then returned to the requesting application, as shown in Figure 4-1 on page 84.

The request is performed by the directory client, and the process that maintains and looks up information in the directory is called the directory server. In general, servers provide a specific service to clients. Sometimes, a server might become the client of other servers in order to gather the information necessary to process a request.

The client and server processes may or may not be on the same machine. A server is capable of serving many clients. Some servers can process client requests in parallel. Other servers queue incoming client requests for serial processing if they are currently busy processing another client's request.

An API defines the programming interface that a particular programming language uses to access a service. The format and contents of the messages exchanged between client and server must adhere to an agreed-upon protocol.

LDAP defines a message protocol used by directory clients and directory servers. There are also associated LDAP APIs for C and Java languages, and ways to access the directory from a Java application using Java Naming and Directory Interface (JNDI). The client is not dependent on a particular implementation of the server, and the server can implement the directory however it chooses.

1.1.4 Distributed directories

The terms *local*, *global*, *centralized*, and *distributed* are often used to describe a directory. These terms mean different things in different contexts. In this section, we explain how these terms apply to directories.

In general, local means nearby, and global means that something is spread across the universe of interest. The universe of interest might be a company, a country, or the Earth. Local and global are two ends of a continuum. That is, something may be more or less global or local than something else. Centralized means that something is in one place, and distributed means that something is in more than one place. As with local and global, something can be distributed to a greater or lesser extent.

The information stored in a directory can be simultaneously local and global in scope. For example, a directory that stores local information might consist of the names, e-mail addresses and so on of members of a department or workgroup. A directory that stores global information might store information for an entire company. Here, the universe of interest is the company.

The clients that access information in the directory can be local or remote. Local clients may all be located in the same building or on the same LAN. Remote clients might be distributed across the continent or planet.

The directory itself can be centralized or distributed. If a directory is centralized, there may be one directory server at one location or a directory server that hosts data from distributed systems. If the directory is distributed, there are multiple servers, usually geographically dispersed, that provide access to the directory.

When a directory is distributed, the information stored in the directory can be partitioned or replicated. When information is partitioned, each directory server stores a unique and non-overlapping subset of the information. That is, each directory entry is stored by one and only one server. One of the techniques to partition the directory is to use LDAP referrals. LDAP referrals enable users to refer LDAP requests to a different server. When information is replicated, the same directory entry is stored by more than one server. In a distributed directory, some information may be partitioned while some may be replicated.

The three *dimensions* of a directory (scope of information, location of clients, and distribution of servers) are independent of each other. For example, clients scattered across the globe can access a directory containing only information about a single department, and that directory can be replicated at many directory servers. Or, clients in a single location can access a directory containing information about everybody in the world that is stored by a single directory server.

The scope of information to be stored in a directory is often given as an application requirement. The distribution of directory servers and the way in which data is partitioned or replicated often can be controlled to affect the performance and availability of the directory.

1.2 Advantages of using a directory

An application-specific directory stores only the information needed by a particular application and is not accessible by other applications. Because a full-function directory service is complex to build, application-specific directories are typically very limited. They probably store only a specific type of information, do not have general search capabilities, do not support replication and partitioning, and probably do not have a full set of administration tools. An application-specific directory could be as simple as a set of editable text files, or it could be stored and accessed in an undocumented, proprietary manner.

In such an environment, each application creates and manages its own application-specific directory, which quickly becomes an administrative nightmare. The same e-mail address stored by the calendar application might also be stored by a mail application and by an application that notifies system operators of equipment problems. Keeping multiple copies of information up-to-date and synchronized is difficult, especially when different user interfaces and even different system administrators are involved.

What is needed is a common, application-independent directory. If application developers could be assured of the existence of a directory service, then application-specific directories would not be necessary. However, a common directory must address the problems mentioned above. It must be based on an open standard that is supported by many vendors on many platforms. It must be accessible through a standard API. It must be extensible so that it can hold the types of data needed by arbitrary applications, and it must provide full functionality without requiring excessive resources on smaller systems. Since more users and applications will access and depend on the common directory, it must also be robust, secure, and scalable.

When such a directory infrastructure is in place, application developers can devote their time to developing applications instead of application-specific directories. In the same way that developers rely on the communications infrastructure of TCP/IP and remote procedure call (RPC) to free them from low-level communication issues, they will be able to rely on powerful, full-function directory services. LDAP is the protocol to be used to access this common directory infrastructure. Like HTTP (hypertext transfer protocol) and FTP (file transfer protocol), LDAP has become an indispensable part of the Internet's protocol suite.

When applications access a standard common directory that is designed in a proper way, rather than using application-specific directories, redundant and costly administration can be eliminated, and security risks are more controllable. For example, the telephone directory, mail, and Web application as shown in Figure 1-1 can all access the same directory to retrieve an e-mail address or other information stored in a single directory entry. The advantage is that the data is kept and maintained in one place. Various applications can use individual attributes of an entry for different purposes permitting that the they have the correct authority. New uses for directory information will be realized, and a synergy will develop as more applications take advantage of the common directory.
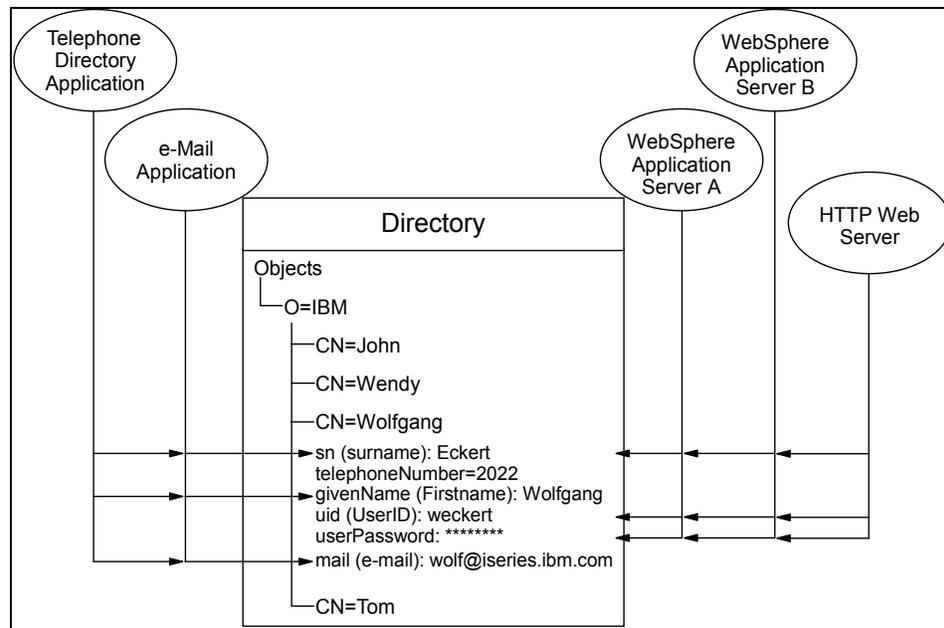
Figure 1-1   Several applications using attributes of the same entry

Storing data in a directory and sharing it amongst applications saves you time and money by keeping administration effort and system resources down. Many IBM applications also utilize directories to centrally store and share information. The number of applications that support LDAP directories is constantly increasing. For example, LDAP directory support, such as for authentication and configuration management, is provided in various IBM operating systems, IBM WebSphere Application Server, IBM WebSphere Portal Server, IBM Tivoli Access Manager, IBM Tivoli Directory Server, IBM HTTP server, IBM Lotus® Domino®, and so forth.

1.3 LDAP history and standards

In the 1970s, the integration of communications and computing technologies led to the development of new communication technologies. Many of the proprietary systems that were developed were incompatible with other systems. It became apparent that standards were needed to allow equipment and systems from different vendors to interoperate. Two independent major standardizations efforts developed to define such standards.

1.3.1 OSI and the Internet

One standards drive was lead by the CCITT (Comite Consultatif International Telephonique et Telegraphique, or International Consultative Committee on Telephony and Telegraphy), and the ISO (International Standards Organization). The CCITT has since become the ITU-T (International Telecommunications Union - Telecommunication Standardization Sector). This effort resulted in the OSI (Open Systems Interconnect) Reference Model (ISO 7498), which defined a seven-layer model of data communication with physical transport at the lower layer and application protocols at the upper layers.

The other standards drive grew up around the Internet and developed from research sponsored by DARPA (the Defense Advanced Research Projects Agency) in the United States. The Internet Architecture Board (IAB) and its subsidiary, the Internet Engineering Task Force (IETF), develop standards for the Internet in the form of documents called Request for Comments (RFCs), which after being approved, implemented, and used for a period of time, eventually become standards (STDs). Before a proposal becomes an RFC, it is called an Internet Draft.

The two standards processes approach standardization from two different perspectives. The OSI approach started from a clean slate and defined standards using a formal committee process without requiring implementations. The Internet uses a less formal engineering approach, where anybody can

propose and comment on RFCs, and implementations are required to verify feasibility.

The OSI protocols developed slowly, and because running the full protocol stack, is resource intensive, they have not been widely deployed, especially in the desktop and small computer market. In the meantime, TCP/IP and the Internet were developing rapidly and being put into use. Also, some network vendors developed proprietary network protocols and products.

1.3.2 X.500 the Directory Server Standard

However, the OSI protocols did address issues important in large distributed systems that were developing in an ad hoc manner in the desktop and Internet marketplace. One such important area was directory services. The CCITT created the X.500 standard in 1988, which became ISO 9594, Data Communications Network Directory, Recommendations X.500-X.521 in 1990, though it is still commonly referred to as X.500.

X.500 organizes directory entries in a hierarchal name space capable of supporting large amounts of information. It also defines powerful search capabilities to make retrieving information easier. Because of its functionality and scalability, X.500 is often used together with add-on modules for interoperation between incompatible directory services.

Note: An excellent online resource on X.500 is the book, Understanding X.500 - The Directory. While dated (1996), this book, which is now out of print (but available online) is considered one of the original "gospels" of the directory world. It describes and defines the X.500 directory model in great detail. Much of the material is still very much relevant in today's current family of LDAP directory servers. It can be found here:

http://www.isi.salford.ac.uk/staff/dwc/X500.htm

X.500 specifies that communication between the directory client and the directory server uses the directory access protocol (DAP). However, as an application layer protocol, the DAP requires the entire OSI protocol stack to operate. Supporting the OSI protocol stack requires more resources than are available in many small environments. Therefore, an interface to an X.500 directory server using a less resource-intensive or lightweight protocol was desired.

1.3.3 Lightweight Access to X.500

LDAP was developed as a lightweight alternative to DAP. LDAP requires the lighter weight and more popular TCP/IP protocol stack rather than the OSI protocol stack. LDAP also simplifies some X.500 operations and omits some esoteric features.

Two precursors to LDAP appeared as RFCs issued by the IETF, Directory Assistance Service (RFC 1202) and DIXIE Protocol Specification (RFC 1249). These were both informational RFCs which were not proposed as standards. The directory assistance service (DAS) defined a method by which a directory client could communicate to a proxy on a OSI-capable host which issued X.500 requests on the client's behalf. DIXIE is similar to DAS, but provides a more direct translation of the DAP.

The first version of LDAP was defined in X.500 Lightweight Access Protocol (RFC 1487), which was replaced by Lightweight Directory Access Protocol (RFC 1777). LDAP further refines the ideas and protocols of DAS and DIXIE. It is more implementation neutral and reduces the complexity of clients to encourage the deployment of directory-enabled applications. Much of the work on DIXIE and LDAP was carried out at the University of Michigan, which provides reference implementations of LDAP and maintains LDAP-related Web pages and mailing lists.

RFC 1777 defines the LDAP protocol itself. RFC 1777, along with:

► The String Representation of Standard Attribute Syntaxes (RFC 1778)

► A String Representation of Distinguished Names (RFC 1779)

► An LDAP URL Format (RFC 1959)

► A String Representation of LDAP Search Filters (RFC 1960)

Define the original LDAPv2 version of the language.

LDAP Version 2 has reached the status of draft standard in the IETF standardization process, one step from being a standard. All of today's directory server implementations are based on the LDAPv3 specification.

LDAP Version 3 is defined by Lightweight Directory Access Protocol (v3) (RFC 2251). Related RFCs that are new or updated for LDAP Version 3 are:

► Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions (RFC 2252)

► Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names (RFC 2253)

► The String Representation of LDAP Search Filters (RFC 2254)

- ► The LDAP URL Format (RFC 2255)
- ► A Summary of the X.500(96) User Schema for use with LDAPv3 (RFC 2256)
- ► Authentication Methods for LDAP (RFC 2829)
- ► LDAPv3: Extension for Transport Layer Security (RFC 2830)
- ► Lightweight Directory Access Protocol (v3): Technical Specification (RFC 3377)

RFC 2251 is a proposed standard, one step below a draft standard. LDAP V3 extended LDAP V2 in the following areas:

- ► Referrals: A server that does not store the requested data can refer the client to another server.

- ► Security: Extensible authentication using Simple Authentication and Security Layer (SASL) mechanism.

- ► Internationalization: UTF-8 support for international characters.

- ► Extensibility: New object types and operations can be dynamically defined and schema published in a standard manner.

In this book, the term LDAP refers to LDAP Version 3 unless LDAP Version 2 is specifically stated. Differences between LDAP Version 2 and LDAP Version 3 are noted when necessary.

1.3.4 Beyond LDAPv3

Recently, the push for encapsulating LDAP operations within XML for use within Web Services has spawned a new language called the Directory Services Markup Language (DSML). The most recent of the specification is DSMLv2. DSML is an XML schema for representing directory information, it's a generic import / export format for directory information. Directory information in DSML can be shared between DSML-aware applications without exposing the LDAP protocol.

XML provides an effective way to present and transfer data; Directory services allow you to share and manage data, and are thus a necessary prerequisite for conducting online business; DSML is designed to make directory service more dynamic by employing XML. DSML is an XML schema for working with directories, it is defined using a Document Content Description (DCD). Thus, DSML allows XML programmers to access LDAP-enabled directories without having to write to the LDAP interface or use proprietary directory-access APIs, and provides one consistent way to work with multiple dissimilar directories

More information on DSML can be found in Appendix A, "DSML Version 2" on page 635.

Various directory integration technologies have emerged in recent years that utilize LDAP and directory concepts to centralize and/or sychronize data between disparate directories as well as other disparate non-directory data sources. Two of the more prominent technologies in this directory integration space are Meta-Directories and Virtual Directories. These technologies are covered in greater detail in Appendix B, "Directory Integration - IBM Tivoli Directory Integrator" on page 681.

1.4 Directory components

A directory contains a collection of objects organized in a tree structure. The LDAP naming model defines how entries are identified and organized. Entries are organized in a tree-like structure called the Directory Information Tree (DIT). Entries are arranged within the DIT based on their distinguished name (DN). A DN is a unique name that unambiguously identifies a single entry. DNs are made up of a sequence of relative distinguished names (RDNs). Each RDN™ in a DN corresponds to a branch in the DIT leading from the root of the DIT to the directory entry. A DN is composed of a sequence of RDNs separated by commas, such as cn=thomas,ou=itso,o=ibm.

You can organize entries, for example, after organizations and within a single organization you can further split the tree into organizational units, and so forth. You can define your DIT based on your organizational needs as shown in Figure 1-2 on page 17. If you have, for example, one company with different divisions, you may want to start with your company name under the root as the organization (o) and then branch into organizational units (ou) for the individual divisions. In case you store data for multiple organizations within a country, you may want to start with a country (c) and then branch into organizations. For more information on planning a DIT, refer to Chapter 3, "Planning your directory" on page 57.

```
+-----------------------------------------------------------------------+
|                        Directory Root (Top)                           |
|                                                                       |
|                 o=IBM                          c=us                   |
|                                                                       |
|       ou=Marketing      ou=Support      o=ACMESupply     o=iSeriesShop |
|                                                                       |
|   cn=mbarlen                         cn=tbarlen                        |
|    objectClass=Person                 objectClass=Person              |
|    objectClass-ePerson                objectClass=ePerson             |
|    mail=marion@ibm.com                mail=thomas@acme.com            |
|    sn=Barlen                          sn=Barlen                       |
|    givenName=Marion                                                   |
|    telephoneNumber=112               deviceID=PrinterSales            |
|                                       objectClass=cimPrinter          |
|   cn=Klaus                            objectClass=ePrinter            |
|    objectClass=Person                 location=Printer room 3rd floor |
|    objectClass=ePerson                owner=John Doe                  |
|    mail=Ktebbe@ibm.com                Queuename=lsprt01               |
|    sn=Tebbe                           maxCopies=10                     |
+-----------------------------------------------------------------------+
```

Figure 1-2   Example of a Directory Information Tree (DIT)

Each object also referred to as an entry in a directory belonging to one or more object classes. An object class describes the content and purpose of the object. It also contains a list of attributes, such as a telephone number or surname, that can be defined in an object of that class. You can publish entries of different object classes under another object as shown in Figure 1-2 where an ePrinter object and a Person object is published under the organization ACMESupply.
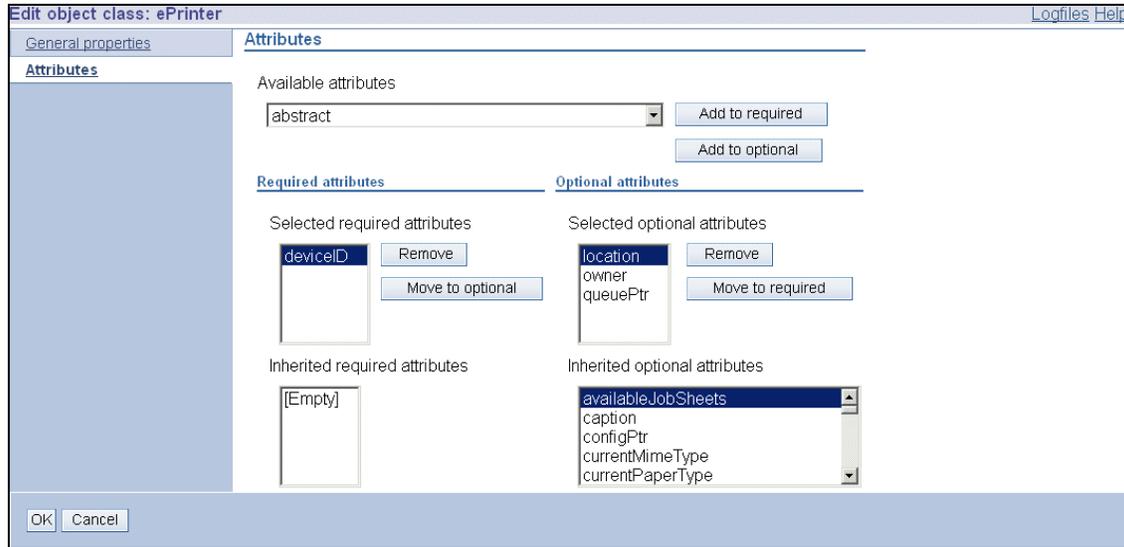
Figure 1-3   ePrinter object class

The object class also defines which of the attributes must be defined (required) when creating an object of this class and which attributes are optional. As shown in Figure 1-3, the object class with the name `ePrinter` has a required attribute `deviceID` and three optional attributes that may or may not be filled in when creating an `ePrinter` object. Object classes can also inherit characteristics, such as attributes from other object classes. In the example of the `ePrinter`, the class inherits all the attributes that are defined in class `cimPrinter`. That means, when you create an `ePrinter` object you have to define the `deviceID` and optionally you can specify the `location`, `owner`, and `queuePtr` attribute of `ePerson` and all attributes of `cimPrinter`.

Also attributes themselves have certain characteristics as shown in Figure 1-4 on page 19. The `surname` attribute name, for example, is defined as `sn` and `surName`, and describes a person's family name. The attribute definition specifies also the syntax rules for the attribute value. A telephone number may only contain numbers and hyphens while the `surname` consists of alpha characters. Other specifications include whether this attribute can contain only one or many values, the matching rules, the Object Identifier (OID), and so forth. The IBM Tivoli Directory Server (ITDS) product also includes some IBM proprietary extensions to each attribute. Other manufactures, such as Microsoft, have similar extensions. The IBM extensions include also an access class, which is used in combination with access control lists (ACLs) to control who can perform a certain action on the attribute value, such as read, write, search, or compare operations.

All the objects and attributes with their characteristics are defined in schemas. The schema specifies what can be stored in the directory. Schema-checking ensures that all required attributes for an entry are present before an entry is stored. Schema-checking also ensures that attributes not in the schema are not stored in the entry. Optional attributes can be filled in at any time. A schema also defines the inheritance and subclassing of objects and where in the DIT structure (hierarchy) objects may appear. Information about the ITDS schema can be found at:

http://publib.boulder.ibm.com/tividd/td/IBMDS/IDSschema52/en_US/HTML/schema.html
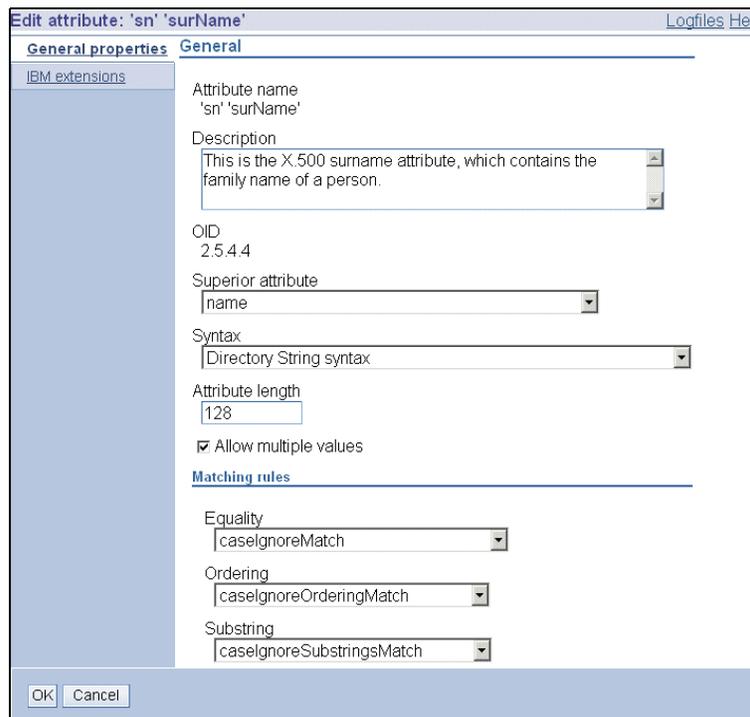
Figure 1-4   Attribute definition example

As you have seen in Figure 1-3 on page 18 and Figure 1-4, object classes and attributes including their specifications are defined as OIDs in an ASN.1 notation format. All these OIDs are registered with a public organization, such as the ANSI organization (http://www.ansi.org) for the United States. The number notation refers to a hierarchy. For example, the OID 2.5.4.4 resolves into a surName attribute as shown in Figure 1-5 on page 20.
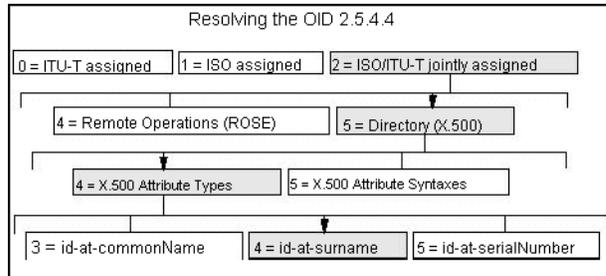
Figure 1-5   Example of object identifiers as defined by the ANSI organization

1.5 LDAP standards

Several standards in the form of IETF RFCs exist for LDAP. The following is a brief list of RFCs that apply for LDAP Version 2 and Version 3:

► RFC 1274 The COSINE and Internet X.500 Schema

► RFC 1777 Lightweight Directory Access Protocol (V2)

► RFC 1778 String Representation of Standard Attribute Syntaxes

► RFC 1779 String Representation of Distinguished Names

► RFC 1823 LDAP Application Program Interface (V2)

► RFC 2052 A DNS RR for Specifying the Location of Services (DNS SRV)

► RFC 2219 Use of DNS Aliases for Network Services

► RFC 2222 Simple Authentication and Security Layer (SASL)

► RFC 2247 Using Domains in LDAP/X.500 Distinguished Names

► RFC 2251 Lightweight Directory Access Protocol (V3)

► RFC 2252 Lightweight Directory Access Protocol (V3): Attribute Syntax Definitions

► RFC 2253 Lightweight Directory Access Protocol (V3): UTF-8 String Representation of Distinguished Names

► RFC 2254 The String Representation of LDAP Search Filters

► RFC 2255 The LDAP URL Format

► RFC 2256 A Summary of the X.500(96) User Schema for use with LDAPv3

► RFC 2596 Use of Language code in LDAP

► RFC 2696 LDAP Control Extension for Simple Paged Results Manipulation

► RFC 2829 Authentication Methods for LDAP

- ► RFC 2849 The LDAP Data Interchange Format (LDIF) - Technical Specification
- ► RFC 2891 LDAP Control Extension for Server Side Sorting of Search Results
- ► The Open Group schema for IiPerson and IiOrganization (NAC/LIPS)
- ► Oasis Directory Services Markup Language (DSML) 2.

1.6 IBM's Directory-enabled offerings

Many of IBM's products are directory enabled in one way or another. Some products have their own LDAP server component (that is, they can respond to queries from LDAP clients), some products require that an LDAP directory exist for them to work at all, and finally some products optionally can take advantage of a LDAP based directory service.

IBM Tivoli Directory Server (ITDS)

ITDS is IBM's LDAPv3 Directory offering. ITDS implements the Internet Engineering Task Force (IETF) LDAP V3 specifications. It also includes enhancements added by IBM in functional and performance areas. This version uses IBM DB2® as the backing store to provide per LDAP operation transaction integrity, high performance operations, and on-line backup and restore capability. ITDS interoperates with the IETF LDAP V3 based clients. Please refer to Chapter 4, "IBM Tivoli Directory Server overview" on page 83, for a more detailed overview of ITDS.

IBM Lotus Domino

IBM Lotus Domino is an enterprise-class messaging and collaboration system, designed to take full advantage of the e-business revolution. It runs on a variety of different hardware platforms and operating systems. IBM Lotus Domino server supports industry standards like Simple Mail Transfer Protocol (SMTP), Multipurpose Internet Mail Extensions (MIME), Post Office Protocol (POP3), LDAP, and SSL.

IBM Lotus Domino is designed to simplify integration into a multi-directory environment. With IBM Lotus Domino (Domino) 6 (or later), you have the option of moving from a distributed directory architecture and making Domino the central directory. This allows you to take advantage of a centralized directory configuration that provides added control and less overhead and is easier to manage. Domino Server comes with the Domino Upgrade Services tool. This tool is used to import users from a server-based foreign directory and register those users in the Domino Directory. Domino Upgrade Services migrates data from many different systems, some of which include LDAP Data Interchange

Format (LDIF) files, LDAP-compliant foreign directories (such as IBM Tivoli Directory Server), Microsoft Windows® NT Server, and Microsoft Active Directory.

IBM Lotus Domino 6.5 also has enhanced the implementation of LDAP capabilities and improved the performance of LDAP directory access. A new Domino LDAP Schema database allows you to maintain and extend the schema.

Other directory schemas can be imported via LDIF files.

Other Domino R6 features include:

► Support for X.500 naming conventions, including hierarchical naming and extensible attributes, for maximum flexibility in configuring the namespace.

► LDAP protocol support in both the client and the server providing lookup (read), add, delete, and modify (write) support for non-Notes clients (for example Web browsers) and servers.

► Rule-based domain relationships for faster lookups across large namespaces.

► Hierarchical naming and trust between domains to support the relationship of entries across domains.

► Support for a Public Key Infrastructure.

► A dynamically extensible directory schema ideal for customizing the directory to meet specific business requirements.

► Multi-master replication, a key element for reliable directory synchronization and maximum availability.

► The LDAP service schema support for LDAP RFCs 2252, 2256, 2798, 2247, 2739, 2079, 1274; the new Domino LDAP Schema database (SCHEMA.NSF) used as a tool for maintaining and extending the schema; an automatic schema maintenance process, true object class inheritance; faster schema loading; and support for the `namingContext` operational attribute defined in LDAP standard RFC 2251.

► An open architecture that can easily incorporate support for emerging standards.

IBM Tivoli Directory Integrator (ITDI)

With the Version 5.2 release of ITDI, ITDI now has the capability, via its LDAP Event Handler, to act as a pseudo LDAP directory server and handle LDAP transactions from various LDAP enabled clients. While ITDI is primarily a meta-directory data synchronization product, the ability to act as an LDAP server can be very useful in many integration scenarios.

ITDI synchronizes identity data residing in directories, databases, collaborative systems, applications used for human resources (HR), customer relationship management (CRM), and Enterprise Resource Planning (ERP), and other corporate applications.

By serving as a flexible, synchronization layer between a company's identity structure and the application sources of identity data, ITDI eliminates the need for a centralized datastore. For those enterprises who do choose to deploy an enterprise directory solution, ITDI can help ease the process by connecting to the identity data from the various repositories throughout the organization.

Please refer to Appendix B, "Directory Integration - IBM Tivoli Directory Integrator" on page 681, for more information about ITDI.

IBM software products that require a directory

These are:

► IBM Tivoli Access Manager

► IBM Tivoli Identity Manager

► IBM Tivoli Privacy Manager

► IBM WebSphere Portal Server

► IBM Lotus Sametime® Server

IBM software products that can take advantage of a directory

These are:

► IBM WebSphere Application Server

► IBM DB2 Universal Database™

► IBM Lotus Notes® Client

1.7 Directory resources on the Web

OpenLDAP is a very active open source LDAPv3 directory server (and associated client tools) project that has been around since 1998. It is derived from the original University of Michigan `slapd` server. The OpenLDAP suite includes:

► Stand-alone LDAP server (`slapd`)

► Stand-alone LDAP replication server (`slurpd`)

► Libraries implementing the LDAP protocol

► Utilities, tools, and sample clients

The OpenLDAP site is also the home of a the JLDAP Java LDAP Class Libraries and the JDBC-LDAP LDAP Bridge Driver.

http://www.openldap.org

The Apache Directory Project is a new Open Source project that is developing an embeddable Java based LDAPv3 directory server.

http://incubator.apache.org/directory/subprojects/eve/index.html

The University of Michigan LDAP Mailing List (ldap@umich.edu mail list) is a popular vendor neutral site used by LDAP developers and system administrators to resolve questions relating to use of LDAP. You can subscribe to the mailing list using the following information

```
SMTP Address: dap-request@umich.edu
subject=SUBSCRIBE
```

Recent messages are archived and can be access directly at:

http://listserver.itd.umich.edu/cgi-bin/lyris.pl?visit=ldap

The LDAPZone is a general purpose site dedicated to directory issues. It has a number of useful forums dealing with development and directory administration.

http://www.ldapzone.com/

The Directory Interoperability Forum (DIF) is the Open Group's directory related working group focused on promotion of directory standards and standard compliance certification.

http://www.opengroup.org/dif/

The Mozilla site contains a number of LDAP SDKs that have been popular since the early days of LDAP development. These include the LDAP C SDK, the Mozilla Java SDK, and PerLDAP.

http://www.mozilla.org/directory/

Net::LDAP is a pure Perl LDAP module available from CPAN. It is actively maintained and provides the most comprehensive set of capabilities for accessing LDAP directories via Perl.

http://search.cpan.org/~gbarr/perl-ldap-0.31/

The Java Naming and Directory Interface (JNDI) is a standard component of Java. It provides the components required to build directory-enabled applications in Java.

http://java.sun.com/products/jndi/

The Active Directory Service Interfaces (ADSI) provides Microsoft based applications the ability to query and manipulate directories.

http://www.microsoft.com/windows2000/techinfo/howitworks/activedirectory/adsilinks.asp

The DirectoryMark is a benchmarking suite designed to measure the performance of directory servers.

http://www.mindcraft.com/directorymark/index.html

The Java LDAP Browser is a very good cross platform (pure Java) LDAP Browser/Editor. It is available for download at:

http://www.iit.edu/~gawojar/ldap/index.html

JXplorer is another good cross platform (pure Java) LDAP Browser/Editor. It also includes very good support for SSL-based LDAP connections.

http://pegacat.com/jxplorer/

2

LDAP concepts and architecture

LDAP is based on the client/server model of distributed computing. The success of LDAP has been largely due to the following characteristics that make it simpler to implement and use, compared to X.500 and DAP.

This chapter explains the basic architecture of LDAP. It discusses the information, naming, functional, and security models that form the basis of the LDAP architecture. Various terms and concepts defined by or needed to understand the LDAP architecture are introduced along the way. After a general overview of the architecture, each of the models that form the backbone of the LDAP architecture is discussed in detail.

2.1 Overview of LDAP architecture

LDAP defines the content of messages exchanged between an LDAP client and an LDAP server. The messages specify the operations requested by the client (that is, search, modify, and delete), the responses from the server, and the format of data carried in the messages. LDAP messages are carried over TCP/IP, a connection-oriented protocol, so there are also operations to establish and disconnect a session between the client and server.

However, for the designer of an LDAP directory, it is not so much the structure of the messages being sent and received over the wire that is of interest. What is important is the logical model that is defined by these messages and data types, how the directory is organized, what operations are possible, how information is protected, and so forth.

The general interaction between an LDAP client and an LDAP server takes the following form:

1. The client establishes a session with an LDAP server. This is known as *binding* to the server. The client specifies the host name or IP address and TCP/IP port number where the LDAP server is listening.

2. The client can provide a user name and a password to properly authenticate with the server, or the client can establish an anonymous session with default access rights. The client and server can also establish a session that uses stronger security methods such as encryption of data.

3. The client then performs operations on directory data. LDAP offers both read and update capabilities. This allows directory information to be managed as well as queried. LDAP also supports searching the directory for data meeting arbitrary user-specified criteria. Searching is a very common operation in LDAP. A user can specify what part of the directory to search and what information to return. A search filter that uses Boolean conditions specifies what directory data matches the search.

4. When the client is finished making requests, it closes the session with the server. This is also known as *unbinding*.

The philosophy of the LDAP API is to keep simple things simple. This means that adding directory support to existing applications can be done with low overhead. Because LDAP was originally intended as a lightweight alternative to DAP for accessing X.500 directories, it follows a X.500 model. The directory stores and organizes data structures known as entries. A directory entry usually describes an object such as a person, device, a location, and so on. Each entry has a name called a distinguished name (DN) that uniquely identifies it. The DN consists of a sequence of parts called relative distinguished names (RDNs), much like a file name consists of a path of directory names in many operating systems such as

UNIX® and Windows. The entries can be arranged into a hierarchical tree-like structure based on their distinguished names. This tree of directory entries is called the Directory Information Tree (DIT).

Each entry contains one or more attributes that describe the entry. Each attribute has a type and a value. For example, the directory entry for a person might have an attribute called *telephoneNumber*. The syntax of the telephoneNumber attribute would specify that a telephone number must be a string of numbers that can contain spaces and hyphens. The value of the attribute would be the person's telephone number, such as 512-555-1212.

A directory entry describes some object. An object class is a general description, sometimes called a template, of an object, as opposed to the description of a particular object. For instance, the object class person has a *surname* attribute, whereas the object describing John Smith has a surname attribute with the value Smith. The object classes that a directory server can store and the attributes they contain are described by schema. Schema define what object classes are allowed where in the directory, what attributes they must contain, what attributes are optional, and the syntax of each attribute. For example, a schema could define a person object class. The person schema might require that a person have a surname attribute that is a character string, specify that a person entry can optionally have a telephoneNumber attribute that is a string of numbers with spaces and hyphens, and so on.

LDAP defines operations for accessing and modifying directory entries such as:

► Binding and unbinding

► Searching for entries meeting user-specified criteria

► Adding an entry

► Deleting an entry

► Modifying an entry

► Modifying the distinguished name or relative distinguished name of an entry (move)

► Comparing an entry

The version of LDAP all modern directory servers use today is LDAPv3. LDAPv3 is documented in several IETF RFCs. The key LDAP Version 3 RFCs are listed below along with a short description to provide an overview of the documents defining the LDAP architecture.

► RFC 2251 *Lightweight Directory Access Protocol* (v3)

Describes the LDAP protocol designed to provide lightweight access to directories supporting the X.500 model. The lightweight protocol is meant to

be implementable in resource-constrained environments such as browsers and small desktop systems.

This RFC is the core of the LDAP family of RFCs. It describes how entries are named with distinguished names, defines the format of messages exchanged between client and server, enumerates the operations that can be performed by the client, and specifies that data is represented using UTF-8 character encoding. The RFC specifies that the schema describing directory entries must themselves be readable so that a client can determine what type of objects a directory server stores. It defines how the client can be referred to another LDAP server if a server does not contain the requested information. It describes how individual operations can be extended using controls and how additional operations can be defined using extensions. It also discusses how clients can authenticate to servers and optionally use Simple Authentication and Security Layer (SASL) to allow additional authentication mechanisms.

► RFC 2252 *Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions*

LDAP uses octet strings to represent the values of attributes for transmission in the LDAP protocol. This RFC defines how values such as integers, time stamps, mail addresses, and so on are represented. For example, the integer 123 is represented by the string "123". These definitions are called attribute syntaxes. This RFC describes how an attribute with a syntax such as "telephone number" is encoded. It also defines matching rules to determine if values meet search criteria. An example is caseIgnoreString, which is used to compare character strings when case is not important.

These attribute types and syntaxes are used to build schema that describe objects classes. A schema lists what attributes a directory entry must or may have. Every directory entry has an objectclass attribute that lists the (one or more) schema that describe the entry. For example, a directory entry could be described by the object classes inetOrgPerson and organizationalPerson. If an objectclass attribute includes the value extensibleObject, it can contain any attribute.

► RFC 2253 *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*

Distinguished names (DNs) are the unique identifiers, sometimes called primary keys, of directory entries. X.500 uses ASN.1 to encode distinguished names. LDAP encodes distinguished names as strings. This RFC defines how distinguished names are represented as strings. A string representation is easy to encode and decode and is also human readable. A DN is composed of a sequence of relative distinguished names (RDNs) separated by commas. The sequence of RDNs making up a DN names the ancestors of a directory entry up to the root of the DIT. Each RDN is composed of an attribute value from the directory entry. For example, the DN cn=John

Smith,ou=Austin,o=IBM,c=US represents a directory entry for a person with the common name (cn) John Smith under the organizational unit (ou) Austin in the organization (o) IBM in the country (c) US.

► RFC 2254 *The String Representation of LDAP Search Filter*s

LDAP search filters provide a powerful mechanism to search a directory for entries that match specific criteria. The LDAP protocol defines the network representation of a search filter. This document defines how to represent a search filter as a human-readable string. Such a representation can be used by applications or in program source code to specify search criteria. Attribute values are compared using relational operators such as equal, greater than, or "sounds like" for approximate or phonetic matching. Boolean operators can be used to build more complex search filters. For example, the following search filter searches for entries that either have a surname attribute of Smith or that have a common name attribute that begins with Jo:

```
(| (sn=Smith) (cn=Jo*))
```

► RFC 2255 *The LDAP URL Format*

Uniform Resource Locators (URLs) are used to identify Web pages, files, and other resources on the Internet. An LDAP URL specifies an LDAP search to be performed at a particular LDAP server. An LDAP URL represents in a compact and standard way the information returned as the result of the search. The LDAP URL Format is discussed in more detail later in this chapter.

► RFC 2256 *A Summary of the X.500(96) User Schema for use with LDAPv3*

Many schema and attributes commonly accessed by directory clients are already defined by X.500. This RFC provides an overview of those attribute types and object classes that LDAP servers should recognize. For instance, attributes such as cn (common name), description, and postalAddress are defined. Object classes such as country, organizationalUnit, groupOfNames, and applicationEntity are also defined.

The RFCs listed above build up the core LDAP Version 3 specification. LDAP can be better understood by considering the four models upon which it is based:

► Information: Describes the structure of information stored in an LDAP directory

► Naming: Describes how information in an LDAP directory is organized and identified

► Functional: Describes what operations can be performed on the information stored in an LDAP directory

► Security: Describes how the information in an LDAP directory can be protected from unauthorized access

The following sections discuss the four LDAP models.

2.2 The informational model

The basic unit of information stored in the directory is called an entry. Entries represent objects of interest in the real world such as people, servers, organizations, and so on. Entries are composed of a collection of attributes that contain information about the object. Every attribute has a type and one or more values. The type of the attribute is associated with a syntax. The syntax specifies what kind of values can be stored. For example, an entry might have a attribute. The syntax associated with this type of attribute would specify that the values are telephone numbers represented as printable strings optionally followed by keywords describing paper size and resolution characteristics. It is possible that the directory entry for an organization would contain multiple values in this attribute—that is, that an organization or person represented by the entity would have multiple fax numbers. The relationship between a directory entry and its attributes and their values is shown in Figure 2-1.
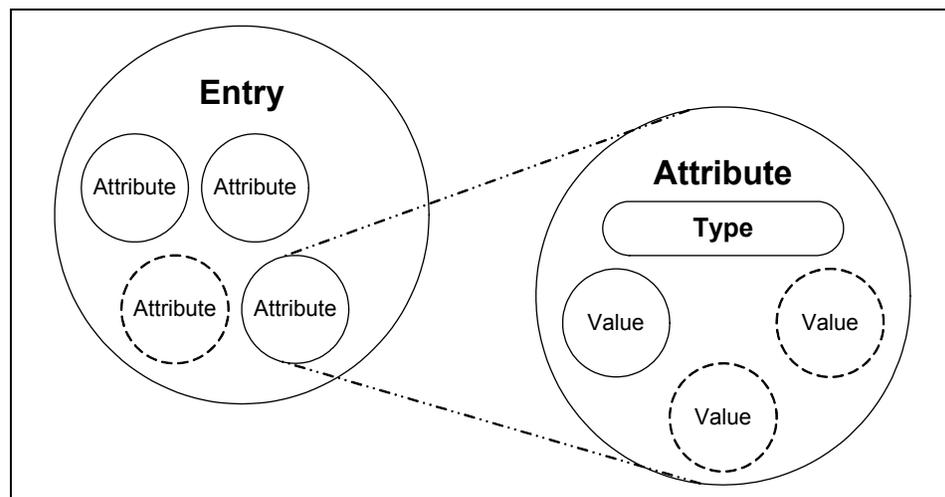
Figure 2-1   Entries, attributes and values

In addition to defining what data can be stored as the value of an attribute, an attribute syntax also defines how those values behave during searches and other directory operations. The attribute telephoneNumber, for example, has a syntax that specifies:

► Lexicographic ordering.
► Case, spaces and dashes are ignored during the comparisons.
► Values must be character strings.

For example, using the correct definitions, the telephone numbers 512-838-6008, 512838-6008, and 5128386008 are considered the same. A few of the syntaxes that have been defined for LDAP are listed in Table 2-1.

Table 2-1   Some of the attribute syntaxes

Syntax	Description
bin	Binary information
ces	Case exact string, also known as a *directory string*, case is significant during comparisons.
cis	Case ignore string. Case is not significant during comparisons.
tel	Telephone number. The numbers are treated as text, but all blanks and dashes are ignored.
dn	Distinguished name.
Generalized Time	Year, month, day, and time represented as a printable string.
Postal Address	Postal address with lines separated by "$" characters.

Table 2-2 lists some common attributes. Some attributes have alias names that can be used wherever the full attribute name is used. For example, cn can be used when referring to the attribute commonName.

Table 2-2   Common LDAP attributes

Attribute, Alias	Syntax	Description	Example
commonName, cn	cis	Common name of an entry	John Smith
surname, sn	cis	Surname (last name) of a person	Smith
telephoneNumber	tel	Telephone number	512-838-6008
organizationalUnit Name, ou	cis	name of organizational unit	Tivoli
owner	dn	DN of person that owns the entry	cn=John Smith,o=IBM,c=us

Attribute, Alias	Syntax	Description	Example
organization, o	cis	Name of organization	IBM
jpegPhoto	bin	Photographic image in JPEG format	Photograph of John Smith

Constraints can be associated with attribute types to limit the number of values that can be stored in the attribute or to limit the total size of a value. For example, an attribute that contains a photo could be limited to a size of 10 KB to prevent the use of unreasonable amounts of storage space. Or an attribute used to store a social security number could be limited to holding a single value.

Schemas define the type of objects that can be stored in the directory. Schemas also list the attributes of each object type and whether these attributes are required or optional. For example, in the person schema, the attribute *surname* (sn) is required, but the attribute *description* is optional. Schema-checking ensures that all required attributes for an entry are present before an entry is stored. Schema-checking also ensures that attributes not in the schema are not stored in the entry. Optional attributes can be filled in at any time. Schema also define the inheritance and subclassing of objects and where in the DIT structure (hierarchy) objects may appear.

Table 2-3 lists a few of the common schema (object classes and their required attributes). In many cases, an entry can consist of more than one object class.

Table 2-3   Object classes and required attributes

Object class	Description	Required attributes
InetOrgPerson	Defines entries for a person	commonName (cn) surname (sn) objectClass
organizationalUnit	Defines entries for organizational units	ou objectClass
organization	Defines entries for organizations	o objectClass

Though each server can define its own schema, for interoperability it is expected that many common schema will be standardized (refer to RFC 2252, *Lightweight Directory Access Protocol (v3): Attribute Syntax Definition*s, and RFC 2256, *A Summary of the X.500(96) User Schema for use with LDAPv3*).

There are times when new schema will be needed at a particular server or within an organization. In LDAP Version 3, a server is required to return information about itself, including the schema that it uses. A program can therefore query a server to determine the contents of the schema. This server information is stored at the special zero-length DN.

Objects can be derived from other objects. This is known as subclassing. For example, suppose an object called *person* was defined that included an attribute *surname* and so on. An object class *organizationalPerson* could be defined as a subclass of the *person* object class. The organizationPerson object class would have the same attributes as the `person` object class and could add other attributes such as *title* and *officenumber*. The `person` object class would be called the superior of the organizationPerson object class. One special object class, called *top*, has no superiors. The top object class includes the mandatory *objectClass* attribute. Attributes in top appear in all directory entries as specified (required or optional).

Each directory entry has a special attribute called objectClass. The value of the objectClass attribute is a list of two or more schema names. These schema define what type of object(s) the entry represents. One of the values must be either top or alias. Alias is used if the entry is an alias for another entry, otherwise top is used. The objectClass attribute determines what attributes the entry must and may have.

The special object class *extensibleObject* allows any attribute to be stored in the entry. This can be more convenient than defining a new object class to add a special attribute to a few entries, but also opens up that object to be able to contain anything (which might not be a good thing in a structured system).

2.2.1 LDIF

When an LDAP directory is loaded for the first time or when many entries have to be changed at once, it is not very convenient to change every single entry on a one-by-one basis. For this purpose, LDAP supports the LDAP Data Interchange Format (LDIF) that can be seen as a convenient, yet necessary, data management mechanism. It enables easy manipulation of mass amounts of data. See Example 2-1 for the basic form of an LDIF entry.

Example 2-1   Basic form of an LDIF entry

```
dn: <distinguished name>
<attrtype> : <attrvalue>
<attrtype> : <attrvalue>
...
```

A line can be continued by starting the next line with a single space or tab character, for example:

```
dn: cn=John E Doe, o=University of Higher
Learning, c=US
```

Multiple attribute values are specified on separate lines, for example:

```
cn: John E Doe
cn: John Doe
```

If an *attrvalue* contains a non-US-ASCII character, or begins with a space or a colon (:), the *attrtype* is followed by a double colon and the value is encoded in base-64 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 a blank line. Multiple blank lines are considered a logical end-of-file.

Example 2-2 shows a simple LDIF file which contains an organizational unit, *People,* located beneath the organization `ibm.com` in the DIT. The entry of John Smith is the only data entry for People. Further on, there is an organizational unit called marketing. Note that John Smith is a member of the marketing department due to the attribute value pair ou: marketing.

Example 2-2   Example LDIF File with organizational and person entries

```
dn: o=ibm.com
objectclass: top
objectclass: organization
o: ibm.com

dn: ou=People, o=ibm.com
objectclass: organizationalUnit
ou: people

dn: ou=marketing, o=ibm.com
objectclass: organizationalUnit
ou: marketing

dn: cn=John Smith, ou=people, o=ibm.com
objectclass: top
objectclass: organizationalPerson
cn: John Smith
sn: Smith
givenname: John
uid: jsmith
ou: marketing
```

```
ou: people
telephonenumber: 838-6004
```

2.2.2 LDAP schema

In this section we discuss LDAP schema.

Objectclasses

An object class is an LDAP term that denotes the type of object being
represented by a directory entry or record. Some typical object types are person,
organization, organizational unit, domain component and groupOfNames. There
are also object classes that define an object's relationship to other objects, such
as object class top denotes that the object may have subordinate objects under it
in a hierarchical tree structure. Note that some LDAP object classes may be
combined, for example, an object class of organizational unit will most often also
be simultaneously defined as a top object class because it will have entries
beneath it.

An object class is declared as *abstract*, *structural*, or *auxiliary*. An abstract
object class is used as a template for creating other object classes. A directory
entry cannot be instantiated from an abstract object class. Directory entries are
instantiated from structural object classes. An auxiliary object class cannot be
instantiated by itself as a directory entry; it can be attached to directory entries
that are instantiated from structural object classes. Auxiliary object classes
provide a method for extending structural object classes without having to
change the schema definition of a structural class.

LDAP object classes defined sets of standard attributes that are listed as *must*
contain (mandatory attributes) and *may* contain (optional attributes). Different
object classes may prescribe some attributes that overlap, or are redundant with
other object classes. And it is common practice in LDAP directories to use
multiple object classes to define a single directory entry. Most object classes are
defined in a hierarchical order, where one object class is said to "inherit" from
another superior object class. Consider an LDAP object that is defined with the
object classes, as shown in Example 2-3.

Example 2-3   LDAP object definition

```
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
objectclass: eDominoAccount
```

The order shown for the object classes above indicates a hierarchical relationship between these object classes, but not necessarily. The top objectclass is of course at the top of the hierarchy. Most other objectclasses that are not intended to be subordinate to another class should have top as its superior. Not all LDAP directories expect a user record to have the top object class assigned to it, while others require it for using Access Control Lists (ACLs) on the object. The person class is subordinate to the top class and requires that the cn (Common Name) and sn (Surname) attributes be populated, and allows several other optional attributes. The organizationalPerson class inherits from the person class. The inetOrgPerson class inherits from the organizationalPerson class. Now here is the tricky part: The eDominoAccount object class is subordinate to the top class and requires that the sn and userid attributes be populated. Notice that this overlaps with the person object class requirement for the sn attribute. Does this mean that we need to store the sn attribute twice? No, because it is a standard attribute. We will talk more about attributes a little later in this section. Example 2-3 on page 37 illustrates that you cannot necessarily tell the hierarchical relationship of object classes by the order they appear in a list. So then, how do we tell? We tell (or in reality, your LDAP directory interface shows you) by looking at the object class definitions themselves. The methods for defining object classes for LDAP V3 are described in RFC-2251 and RFC-2252. Example 2-4 shows object class definitions taken from ITDS.

Example 2-4   Some ITDS object class definitions

```
objectclass: top
objectclasses=( 2.5.6.0 NAME 'top' DESC 'Standard ObjectClass' ABSTRACT MUST (
objectClass ) )


objectclass: person
objectclasses=( 2.5.6.6 NAME 'person' DESC 'Defines entries that generically
represent people.' SUP 'top' STRUCTURAL MUST ( cn $ sn ) MAY ( userPassword $
telephoneNumber $ seeAlso $ description ) )


objectclass: organizationalPerson

objectclasses=( 2.5.6.7 NAME 'organizationalPerson' DESC 'Defines entries for
people employed by or associated with an organization.' SUP 'person' STRUCTURAL
MAY ( title $ x121Address $ registeredAddress $ destinationIndicator $
preferredDeliveryMethod $ telexNumber $ teletexTerminalIdentifier $
internationalISDNNumber $ facsimileTelephoneNumber $ street $ postalAddress $
postalCode $ postOfficeBox $ physicalDeliveryOfficeName $ ou $ st $ l ) )
objectclass: inetOrgPerson

objectclasses=( 2.16.840.1.113730.3.2.2 NAME 'inetOrgPerson' DESC 'Defines
entries representing people in an organizations enterprise network.' SUP
'organizationalPerson' STRUCTURAL MAY ( audio $ businessCategory $ carLicense $
```

```
departmentNumber $ employeeNumber $ employeeType $ givenName $ homePhone $
homePostalAddress $ initials $ jpegPhoto $ labeledURI $ mail $ manager $ mobile
$ pager $ photo $ preferredLanguage $ roomNumber $ secretary $ uid $
userCertificate $ userSMIMECertificate $ x500UniqueIdentifier $ displayName $ o
$ userPKCS12 ) )
```

Note that each object class begins with a string of numbers delimited by decimals. This number is referred to as the OID (object identifier). After the OID is the object class name (NAME) followed by a description (DESC). If it is subordinate to another object class, the superior (SUP) object class is listed. Finally, the object class definition specifies what attributes are mandatory (MUST) and which are optional (MAY).

The OID is a numeric string that is used to uniquely identify an object. OIDs are a managed hierarchy administered by the International Organization for Standardization (ISO - Web site http://www.iso.ch/) and the International Telecommunication Union (ITU - Web site http://www.itu.ch/). ISO and ITU delegate OID management to organizations by assigning them OID numbers. Organizations can then assign OIDs to objects or further delegate to other organizations. OIDs are associated with objects in protocols and data structures defined using Abstract Syntax Notation (ASN.1).

OIDs are intended to be globally unique. They are formed by taking a unique numeric string (for example, 1.3.4.7.4.17) and adding additional digits in a unique fashion (such as 1.3.4.7.4.17.1, 1.3.4.7.4.17.2, 1.3.4.7.4.17.3, etc.) An organization may acquire a "branch" from some root or vertex in the OID tree. Such a branch is more commonly referred to as an arc (in the previous example it was 1.3.4.7.4.17). The organization may then extend the arc (called subarcs) as shown above to create additional OIDs and arcs. We have no idea why the terminology for the OID tree uses the words "vertex" and "arc" instead of "root" and "branch" as is more commonly used in LDAP and its X.500 heritage.

If you have an LDAP directory that is a derivative of the original University of Michigan LDAP code (many open source and commercial LDAP directory servers are), your object class definitions are contained in files ending with ".oc".

Note that IBM-specific OIDs begin with the arc 1.3.18.0.2; this is a unique private enterprise number that has been assigned to IBM. The number breaks down as shown in Example 2-5.

Example 2-5   IBM-specific OIDs

```
1 (ISO-assigned OID)
1.3 (ISO-identified organization)
1.3.18 (IBM)
1.3.18.0 (IBM Objects)
```

As you may have guessed, the "dot notation" as first used by the IETF for IP addresses was adopted to OIDs to keep things simple. However, unlike IP addresses, there is no limit to the length of an OID.

If your organization must define your own attributes for use within your internal directories, you should consider obtaining your own private enterprise number arc to identify these attributes. We do not recommend that you "make up" your own numbers, as you will probably not be able to interoperate with other organizations (or some vendor's LDAP products). This is not to say obtaining your own OID arc from ISO, IANA or some other authority to define your own object classes and attributes will guarantee interoperability. But it will prevent you from using OIDs that have already been assigned to or by someone else. OIDs are only used for "equality-matching". That is, two objects (for example, directory attributes or certificate policies) are considered to be the same if they have exactly the same OID. There are no implied navigational or hierarchical capabilities with OIDs (unlike IP addresses, for example); given an OID one can not readily find out who owns the OID, related OIDs, etc. OIDs exist to provide a unique identifier. There is nothing to stop two organizations from picking the same identical names for objects that they manage, however, the OIDs will be unique assuming they were assigned from legitimate arc numbers. If you are interested in obtaining a private enterprise number (arc) for your own organization, you may apply for one (free of charge) at the Internet Assigned Numbers Authority Web site:

http://www.iana.org/cgi-bin/enterprise.pl

For more information regarding OIDs, the trees of assigned numbers, and registration, we recommend starting at the ASN.1 frequently asked questions Web site at:

http://asn1.elibel.tm.fr/oid/faq.htm

Let us look at the following example: Top is an abstract class that contains the objectClass attribute. Person is a structural class that instantiates the directory entry for a given person where the objectClass attribute is also part of that Person entry. So far, this example has used only attributes and object classes defined in a standard. So, now, you may want to tailor the people entries to include information that your company requires and that is not defined in the standard Person object definition. There are two ways to do this:

► Subclass the Person object to create a new structural class that includes those additional attributes defined by your company, and instantiate the Person directory entry based on that new class.

► Define that collection of company attributes needed for your company's Person definition as an auxiliary class, and attach it to the directory entry instantiated from the Person class.

Either method is recommended. The downside to auxiliary classes is that if the auxiliary class includes an attribute that is also included in the structural class definition, when that attribute is included in the instantiated directory entry and that attribute contains multiple values and you want to delete the attribute, you cannot tell whether the attribute (when added to the entry) was added when the structural class was instantiated or when the auxiliary class was instantiated. This may be important to the implementor or administrator.

Special entries exist in the namespace, called aliases. Aliases represent links to other entries or partitions of the namespace. When the distinguished name of an alias is used, the entry accessed is the entry to which the alias refers (unless specified otherwise through the programming interface). The collection of directory entries forms the Directory Information Tree (DIT). The method of storage for the DIT of the LDAP directory is implementation-dependent and hidden from the user of that LDAP directory. For example, the ITDS uses IBM DB2 as its data store, but no DB2 constructs are externalized to LDAP.

Attributes

All the object class does is define the attributes, or types of data items contained in that type of object. Some examples of typical attributes are cn (common name), sn (surname), givenName, mail, uid, and userPassword. Just as the object classes are defined with unique OIDs, each attribute also has a unique OID number assigned to it. LDAP V3 attributes follow a notation similar (ASN.1) to object classes. Example 2-6 shows some attribute definitions.

Example 2-6   Attribute definitions

```
attribute: name
attributetypes=( 2.5.4.41 NAME 'name' DESC 'The name attribute type is the
attribute supertype from which string attribute types typically used for naming
may be formed. It is unlikely that values of this type itself will occur in an
entry.' EQUALITY 1.3.6.1.4.1.1466.109.114.2 SUBSTR 2.5.13.4 SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )


attribute: sn
attributetypes=( 2.5.4.4 NAME ( 'sn' 'surName' ) DESC 'This is the X.500
surname attribute, which contains the family name of a person.' SUP 2.5.4.41
EQUALITY 2.5.13.2 ORDERING 2.5.13.3 SUBSTR 2.5.13.4 USAGE userApplications )


attribute: mail
attributetypes=( 0.9.2342.19200300.100.1.3 NAME ( 'mail' 'rfc822mailbox' ) DESC
'Identifies a users primary email address (the email address retrieved and
```

```
displayed by white-pages lookup applications).' EQUALITY 2.5.13.2 SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplication)
```

Notice in Example 2-6 on page 41 that the superior (SUP) of sn is the attribute
2.5.4.41, which happens to be the name attribute. But then the name attribute
description says `unlikely that values of this type itself will occur....`
This illustrates just one of the many peculiarities of the way the attributes have
been defined. It merely provides a shorthand way to defining name-like attributes
such as surname. We did not need to define the syntax for sn because it inherits
this from name.

The attribute mail also has an alias of rfc822mailbox. As you may have guessed,
the "EQUALITY" and "SYNTAX" are yet more ASN.1 definitions.

2.3 The naming model

The LDAP naming model defines how entries are identified and organized.
Entries are organized in a tree-like structure called the Directory Information Tree
(DIT). Entries are arranged within the DIT based on their distinguished name
(DN). A DN is a unique name that unambiguously identifies a single entry. DNs
are made up of a sequence of relative distinguished names (RDNs).

Each RDN in a DN corresponds to a branch in the DIT leading from the root of
the DIT to the directory entry. Each RDN is derived from the attributes of the
directory entry. In the simple and common case, an RDN has the form
`<attribute name> = <value>`. A DN is composed of a sequence of RDNs
separated by commas.

An example of a DIT is shown in Figure 2-2 on page 43. The example is very
simple, but can be used to illustrate some basic concepts. Each box represents a
directory entry. The root directory entry is conceptual, but does not actually exist.
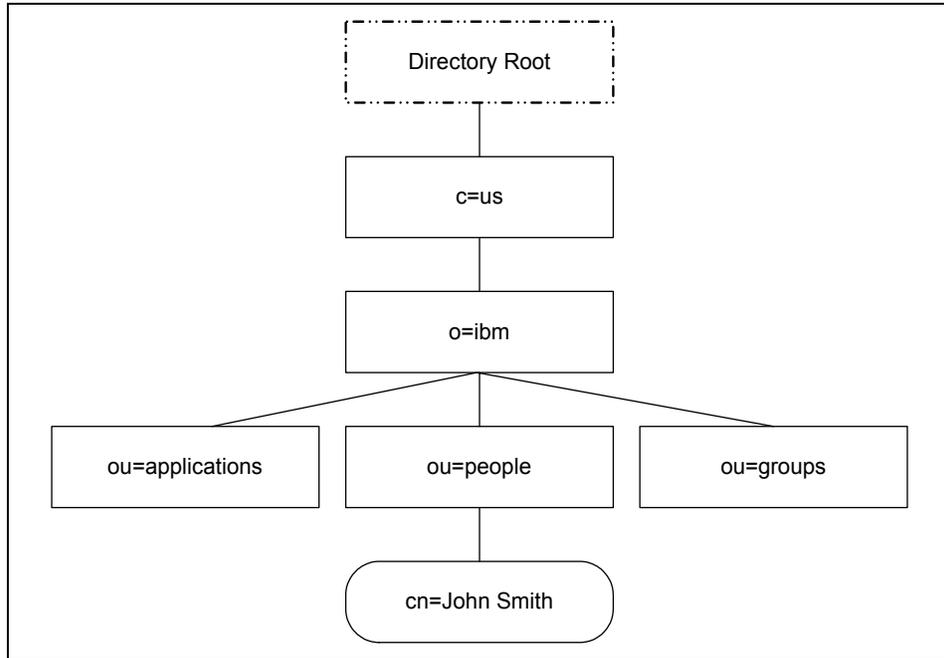
Figure 2-2   Example of a Directory Information Tree (DIT)

The organization of the entries in the DIT is restricted by their corresponding object class definitions.

Entries are named according to their position in the DIT. The directory entry at the bottom of the figure has the DN of cn=John Smith,ou=people,o=ibm,c=us. The organizational group people has the DN of ou=people,o=ibm,c=us.

2.3.1 LDAP distinguished name syntax (DNs)

Entries in an LDAP directory are identified by their names. The characteristics of these names are:

► They have two forms: A string representation and a URL.
► They have a uniform syntax.
► Namespace boundaries are not apparent in them.

A component of a name is called a relative distinguished name (RDN). An RDN represents a point within the namespace hierarchy. RDNs are separated by and concatenated using a comma (,). Each RDN is typed. RDNs have the form *type=value* for single valued RDNs. The plus sign (+) is used to form multi-valued RDNs: *type=value+type=value*.

The *type* is case-insensitive and the *value* is defined to have a particular syntax. The order of RDNs in an LDAP name is the most specific RDN first followed by the less specific RDNs moving up the DIT hierarchy. A concatenated series of RDNs equates to a distinguished name. The DN is used to represent an object and the path to the object in the hierarchical namespace. A URL format for LDAP has been defined that includes a DN as a component of the URL. These forms are explained in the sections that follow.

Every entry in the directory has a 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=Roger Smith,ou=sales,o=ib,c=US
cn=Sandy Brown,ou=marketing,o=ibm,c=US
cn=Leslie Jones,ou=development,o=ibm,c=US
```

Any of the attributes defined in the directory schema may be used to make up a DN. The order of the component attribute value pairs is important. The DN contains one component for each level of the directory hierarchy from the root down to the level where the entry resides. LDAP DNs begin with the most specific attribute (usually some sort of name), and continue with progressively broader attributes, often ending with a country attribute. The first component of the DN is referred to as the Relative Distinguished Name (RDN). It identifies an entry distinctly from any other entries that have the same parent. In the examples above, the RDN cn=Roger Smith separates the first entry from the second entry, (with RDN cn=Sandy Brown). These two example DNs are otherwise equivalent. The attribute:value pair making up the RDN for an entry must also be present in the entry. (This is not true of the other components of the DN.)

The Distinguished Name (DN) syntax supported by this server is based on RFC 2253. The Backus-Naur Form (BNF) syntax is shown in Example 2-7.

Example 2-7   DN syntax

```
<name> ::= <name-component> ( <spaced-separator> )
         | <name-component> <spaced-separator> <name>

  <spaced-separator> ::= <optional-space>
                  <separator>
                  <optional-space>

  <separator> ::=  "," | ";"

  <optional-space> ::= ( <CR> ) *( " " )

  <name-component> ::= <attribute>
          | <attribute> <optional-space> "+"
            <optional-space> <name-component>
```

```
<attribute> ::= <string>
        | <key> <optional-space> "=" <optional-space> <string>

<key> ::= 1*( <keychar> ) | "OID." <oid> | "oid." <oid>
<keychar> ::= letters, numbers, and space

<oid> ::= <digitstring> | <digitstring> "." <oid>
<digitstring> ::= 1*<digit>
<digit> ::= digits 0-9

<string> ::= *( <stringchar> | <pair> )
        | '"' *( <stringchar> | <special> | <pair> ) '"'
        | "#" <hex>


<special> ::= "," | "=" | <CR> | "+" | "<" |  ">"
        | "#" | ";"

<pair> ::= "\" ( <special> | "\" | '"')
<stringchar> ::= any character except <special> or "\" or '"'


<hex> ::= 2*<hexchar>
<hexchar> ::= 0-9, a-f, A-F
```

A semicolon (;) character can be used to separate RDNs in a distinguished name,
although the comma (,) character is the typical notation.

White-space characters (spaces) might be present on either side of the comma or
semicolon. The white-space characters are ignored, and the semicolon is
replaced with a comma.

In addition, space (' ' ASCII 32) characters may be present either before or
after a '+' or '='. These space characters are ignored when parsing.

A value may be surrounded by double quotation ('"' ACSII 34) characters, which
are not part of the value. Inside the quoted value, the following characters
can occur without being interpreted as escape characters:

A space or "#" character occurring at the beginning of the string
A space character occurring at the end of the string
One of the characters "'", "=", "+", "\", "<", ">", or ";"
Alternatively, a single character to be escaped may be prefixed by a backslash
('\' ASCII 92). This method can be used to escape any of the characters listed
previously and the double quotation marks ('"' ASCII 34) character.

This notation is designed to be convenient for common forms of names. The
following example is a distinguished name written using this notation. First is

```
a name containing three components. The first of the components is a multi
valued RDN. A multivalued RDN contains more than one attribute:value pair and
can be used to distinctly identify a specific entry in cases where a simple CN
value might be ambiguous:

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

2.3.2 String form

The exact syntax for names is defined in RFC 2253. Rather than duplicating the
RFC text, the following are examples of valid distinguished names written in
string form:

► cn=Leslie Smith, ou=Austin, o=IBM

  This is a name containing three relative distinguished names (RDNs).

► ou=deptUVZS + cn=Leslie Smith, ou=Austin, o=IBM

  This a name containing three RDNs in which the first RDN is multi-valued.

► cn=L. Eagle, o=Sue\, Grabbit and Runn, c=GB

  This example shows the method of quoting a comma (using a backslash as
  the escape character) in an organization name.

► cn=Before\0DAfter,o=Test,c=GB

  This is an example name in which a value contains a carriage return
  character (0DH).

► sn=Lu\C4\8Di\C4\87

  This last example represents an RDN surname value consisting of five letters
  (including non-standard ASCII characters) that is written in printable ASCII
  characters. Table 2-4 explains the quoted character codes.

Table 2-4   The ASCII encoding of an RDN surname (example)

Unicode letter description	ISO 10646 code	UTF-8	Quoted
Latin capital letter L	U0000004C	0x4C	L
Latin capital letter u	U00000075	0x75	u
Latin small letter c with caron	U0000010D	0xC48D	\C4\8D
Latin small letter i	U00000069	0x69	i
Latin small letter c with acute	U00000107	0xC487	\C4\87

For the detailed definition of DNs in string form, consult RFC 2253. More about Unicode character encoding (superset of ISO 10646) and its transformation into UTF-8 can be found at http://www.unicode.org and in RFC 2279.

2.3.3 URL form

The LDAP URL format has the general form ldap://<host>:<port>/<path>, where <path> has the form <dn>[?<attributes>[?<scope>?<filter>]].

The <dn> is an LDAP distinguished name using a string representation. The <attributes> indicate which attributes should be returned from the entry or entries. If omitted, all attributes are returned. The <scope> specifies the scope of the search to be performed. Scopes may be current entry, one-level (current entry's children), or the whole subtree. The <filter> specifies the search filter to apply to entries within the specified scope during the search. The URL format allows Internet clients, for example, Web browsers, to have direct access to the LDAP protocol and thus LDAP directories.

Examples of LDAP URLs are:

► ldap://austin.ibm.com/ou=Austin,o=IBM

This URL corresponds to a base object search of the <ou=Austin, o=IBM> entry using a filter <of objectClass=*> requesting all attributes (if a filter is omitted, a filter of <objectClass=*> is assumed by definition).

► ldap://austin.ibm.com/o=IBM?postalAddress

This is an LDAP URL referring to only the postalAddress attribute of the IBM entry.

► ldap:///ou=Austin,o=IBM??sub?(cn=Joe Q. Public)

This is an LDAP URL referring to the set of entries found by querying any capable LDAP server (no hostname was given) and doing a subtree search of the IBM Austin subtree for any entry with a common name of Joe Q. Public retrieving all attributes. The LDAP URL format is defined in RFC 2255.

2.4 Functional model

The LDAP functional model is comprised of three categories of operations that can be performed against a LDAPv3 directory service:

► Authentication: *Bind*, *Unbind*, and *Abandon* operations used to connect and disconnect to and from an LDAP server, establish access rights and protect information.

► Query: *Search* for and *Compare* entries for entries meeting user-specified criteria.

► Update: *Add* an entry, *Delete* an entry, *Modify* an entry, and modify the distinguished name (*ModifyRDN*) or relative distinguished name of an entry.

2.4.1 Query

The most common operation is search. The search operation is very flexible and has some of the most complex options.

The search operation allows a client to request that an LDAP server search through some portion of the DIT for information meeting user-specified criteria in order to read and list the result(s). There are no separate operations for read and list; they are incorporated in the search function. The search can be very general or very specific. The search operation allows one to specify the starting point within the DIT, how deep within the DIT to search, what attributes an entry must have to be considered a match, and what attributes to return for matched entries.

Some example searches expressed informally in English are:

► Find the postal address for cn=John Smith,o=IBM,c=DE.

► Find all the entries that are children of the entry ou=ITSO,o=IBM,c=US.

► Find the e-mail address and phone number of anyone in IBM whose last name contains the characters "miller" and who also has a fax number.

To perform a search, the following parameters must be specified:

► Base

A DN that defines the starting point, called the base object, of the search. The base object is a node within the DIT.

► Scope

Specifies how deep within the DIT to search from the base object. There are three choices: *baseObject*, *singleLevel*, and *wholeSubtree*. If baseObject is specified, only the base object is examined. If singleLevel is specified, only the immediate children of the base object are examined; the base object itself is not examined. If wholeSubtree is specified, the base object and all of its descendants are examined.

► Search Filter

Specifies the criteria an entry must match to be returned from a search. The search filter is a Boolean combination of attribute value assertions. An attribute value assertion tests the value of an attribute for equality, less than or equal to, and so on. For example, a search filter might specify entries with a common name containing "wolf" or belonging to the organization ITSO.

► Attributes to Return

Specifies which attributes to retrieve from entries that match the search criteria. Since an entry may have many attributes, this allows the user to only see the attributes they are interested in. Normally, the user is interested in the value of the attributes. However, it is possible to return only the attribute types and not their values. This could be useful if a large value like a JPEG photograph was not needed for every entry returned from the search, but some of the photographs would be retrieved later as needed.

► Alias Dereferencing

Specifies if aliases are dereferenced—that is, if the alias entry itself or the entry it points to is used. Aliases can be dereferenced or not when locating the base object and/or when searching under the base object. If aliases are dereferenced, then they are alternate names for objects of interest in the directory. Not dereferencing aliases allows the alias entries themselves to be examined.

► Limits

Searches can be very general, examining large subtrees and causing many entries to be returned. The user can specify time and size limits to prevent wayward searching from consuming too many resources. The size limit restricts the number of entries returned from the search. The time limit limits the total time of the search. Servers are free to impose stricter limits than requested by the client.

2.4.2 Referrals and continuation references

If the server does not contain the base object, it will return a referral to a server that does, if possible. Once the base object is found singleLevel and wholeSubtree searches may encounter other referrals. These referrals are returned in the search result along with other matching entries. These referrals are called continuation references because they indicate where a search could be continued. For example, when searching a subtree for anybody named Smith, a continuation reference to another server might be returned, possibly along with several other matching entries. It is not guaranteed that an entry for somebody named Smith actually exists at that server, only that the continuation reference points to a subtree that could contain such an entry. It is up to the client to follow continuation references if desired. Since only LDAP Version 3 specifies referrals, continuation references are not supported in earlier versions.

2.4.3 Search filter syntax

The search filter defines criteria that an entry must match to be returned from a search. The basic component of a search filter is an attribute value assertion of the form:

```
attribute operator value
```

For example, to search for a person named John Smith the search filter would be cn=John Smith. In this case, cn is the attribute, = is the operator, and John Smith is the value. This search filter matches entries with the common name John Smith. Table 2-5 shows the search filter options.

Table 2-5   Search filter options

Operator	Description	Example
=	Returns entries whose attribute is equal to the value.	cn=John Smith finds the entry with common name John Smith.
>=	Returns entries whose attribute is greater than or equal to the value.	sn>=smith finds all entries from smith to z*.
<=	Returns entries whose attribute is less than or equal to the value.	sn<=smith finds all entries from a* to smith.
=*	Returns entries that have a value set for that attribute.	sn=* finds all entries that have the sn attribute.
~=	Returns entries whose attribute value approximately matches the specified value. Typically, this is an algorithm that matches words that sound alike.	sn~= smit might find the entry "sn=smith".

The * character matches any substring and can be used with the = operator. For example, cn=J*Smi* would match John Smith and Jan Smitty.

Search filters can be combined with Boolean operators to form more complex search filters. The syntax for combining search filters is:

```
( "&" or "|" (filter1) (filter2) (filter3) ...) ("!" (filter))
```

The Boolean operators are listed in Table 2-6 on page 51.

Table 2-6   Boolean operators

Boolean operator	Description
&	Returns entries matching all specified filter criteria.
|	Returns entries matching one or more of the filter criteria.
!	Returns entries for which the filter is not true. This operator can only be applied to a single filter. (!(filter)) is valid, but (!(filter1)(filter2)) is not.

For example, (|(sn=Smith)(sn=Miller)) matches entries with the surname Smith or the surname Miller. The Boolean operators can also be nested as in (| (sn=Smith) (&(ou=Austin)(sn=Miller))), which matches any entry with the surname Smith or with the surname Miller that also has the organizational unit attribute Austin.

2.4.4 Compare

The compare operation compares an entry for an attribute value. If the entry has that value, compare returns TRUE. Otherwise, compare returns FALSE. Although compare is simpler than a search, it is almost the same as a base scope search with a search filter of attribute=value. The difference is that if the entry does not have the attribute at all (the attribute is not present), the search will return `not found`. This is indistinguishable from the case where the entry itself does not exist. On the other hand, compare will return FALSE. This indicates that the entry does exist, but does not have an attribute matching the value specified.

2.4.5 Update operations

Update operations modify the contents of the directory. Table 2-7 summarizes the update operations.

Table 2-7   Update operations

Operation	Description
add	Inserts new entries into the directory.
delete	Deletes existing entries from the directory. Only leaf nodes can be deleted. Aliases are not resolved when deleting.
modify	Changes the attributes and values contained within an existing entry. Allows new attributes to be added and existing attributes to be deleted or modified.

Operation	Description
modify DN	Changes the least significant (left most) component of a DN or moves a subtree of entries to a new location in the DIT. Entries cannot be moved across server boundaries.

2.4.6 Authentication operations

Authentication operations are used to establish and end a session between an LDAP client and an LDAP server. The session may be secured at various levels ranging from an insecure anonymous session, an authenticated session in which the client identifies itself by providing a password, to a secure, encrypted session using SASL mechanisms. SASL was added in LDAP Version 3 to overcome the weak authentication in LDAP Version 2. Table 2-8 summarizes the authentication operations.

Table 2-8   Authentication operations

Operation	Description
Bind	Initiates an LDAP session between a client and a server. Allows the client to prove its identity by authenticating itself to the server.
Unbind	Terminates a client/server session.
Abandon	Allows a client to request that the server abandon an outstanding operation.

2.4.7 Controls and extended operations

Controls and extended operations allow the LDAP protocol to be extended without changing the protocol itself. Controls modify the behavior of an operation, and extended operations add new operations to the LDAP protocol. The list of controls and extensions supported by an LDAP server can be obtained by examining the root DSE of that server. Controls can be defined to extend any operation.

Controls are added to the end of the operation's protocol message. They are supplied as parameters to functions in the API.

A control has a dotted decimal string object ID used to identify the control, an arbitrary control value that holds parameters for the control, and a criticality level. If the criticality level is TRUE, the server must honor the control; or if the server does not support the control, reject the entire operation. If the criticality level is FALSE, a server that does not support the control must perform the operation as if there was no control specified. For example, a control might extend the delete

operation by causing an audit record of the deletion to be logged to a file specified by the control value information.

An extended operation allows an entirely new operation to be defined. The extended operation protocol message consists of a dotted decimal string object ID used to identify the extended operation and an arbitrary string of operation-specific data.

2.5 Security model

The security model is based on the bind operation. There are several different bind operations possible, and thus the security mechanism applied is different as well. One possibility is when a client requesting access supplies a DN identifying itself along with a simple clear-text password. If no DN and password is declared, an anonymous session is assumed by the LDAP server. The use of clear text passwords is strongly discouraged when the underlying transport service cannot guarantee confidentiality and may therefore result in disclosure of the password to unauthorized parties.

LDAP V3 comes along with a bind command supporting the Simple Authentication and Security Layer (SASL) mechanism. This is a general authentication framework, where several different authentication methods are available for authenticating the client to the server; one of them is Kerberos.

Furthermore, extended protocol operations are available in LDAP V3. An extension related to security is the Extension for Transport Layer Security (TLS) for LDAPv3. This allow operations too use TLS as a means to encrypt an LDAP session and protect against spoofing. TLS has a mechanism which enables it to communicate to an SSL server so that it is backwards compatible. The basic principles of SSL and TLS are the same.

2.6 Directory security

Security is of great importance in the networked world of computers, and this is true for LDAP as well. When sending data over insecure networks, internally or externally, sensitive information may need to be protected during transportation. There is also a need to know who is requesting the information and who is sending it. This is especially important when it comes to the update operations on a directory. The term security, as used in the context of this book, generally covers the following four aspects:

► Authentication: Assurance that the opposite party (machine or person) really is who he/she/it claims to be.

- ► Integrity: Assurance that the information that arrives is really the same as what was sent.

- ► Confidentiality: Protection of information disclosure by means of data encryption to those who are not intended to receive it.

- ► Authorization: Assurance that a party is really allowed to do what he/she/it is requesting to do. This is basically achieved by assigning access controls, like read, write, or delete, to user IDs or common names.

The following sections focus on the first three aspects (since authorization is not yet contained in the LDAP Version 3 standard): Authentication, integrity and confidentiality. There are several methods that can be used for this purpose; the most important ones are discussed here. These are:

- ► No authentication.

- ► Basic authentication.

- ► Simple Authentication and Security Layer (SASL). This includes DIGEST-MD5. When a client uses Digest-MD5, the password is not transmitted in clear text and the protocol prevents replay attacks.

2.6.1 No authentication

This is the simplest authentication method, one that obviously does not need to be explained in much detail. This method should only be used when data security is not an issue and when no special access control permissions are involved. This could be the case, for example, when your directory is an address book browsable by anybody. No authentication is assumed when you leave the password and DN fields empty in an ldap operation. The LDAP server then automatically assumes an anonymous user session and grants access with the appropriate access controls defined for this kind of access (not to be confused with the SASL anonymous user).

2.6.2 Basic authentication

The security mechanism in LDAP is negotiated when the connection between the client and the server is established. This is the approach specified in the LDAP application program interface (API). Besides the option of using no authentication at all, the most simple security mechanism in LDAP is called basic authentication, which is also used in several other Web-related protocols, such as in HTTP.

When using basic authentication with LDAP, the client identifies itself to the server by means of a DN and a password which are sent in the clear over the network (some implementations may use Base64 encoding instead). The server

considers the client authenticated if the DN and password sent by the client match the password for that DN stored in the directory. Base64 encoding is defined in the Multipurpose Internet Mail Extensions (MIME) LDAP standard (RFC 1521). It is a relatively simple encryption, and therefore it is not hard to break once one has captured the data on the network.

2.6.3 SASL

SASL is a framework for adding additional authentication mechanisms to connection-oriented protocols. It has been added to LDAP Version 3 to overcome the authentication shortcomings of Version 2. SASL was originally devised to add stronger authentication to the IMAP protocol. SASL has since evolved into a more general system for mediating between protocols and authentication systems. It is a proposed Internet standard defined in RFC 2222.

In SASL, connection protocols, like LDAP, IMAP, and so on, are represented by profiles; each profile is considered a protocol extension that allows the protocol and SASL to work together. A complete list of SASL profiles can be obtained from the Information Sciences Institute (ISI). Each protocol that intends to use SASL needs to be extended with a command to identify an authentication mechanism and to carry out an authentication exchange. Optionally, a security layer can be negotiated to encrypt the data after authentication and so ensure confidentiality. LDAP Version 3 includes a command (ldap_sasl_bind()) to encrypt the data after authentication.

2.6.4 SSL and TLS

The Secure Socket Layer (SSL) protocol was devised to provide both authentication and data security. It encapsulates the TCP/IP socket so that basically every TCP/IP application can use it to secure its communication.

SSL/TLS supports server authentication (client authenticates server), client authentication (server authenticates client), or mutual authentication. In addition, it provides for privacy by encrypting data sent over the network. SSL/TLS uses a public key method to secure the communication and to authenticate the counterparts of the session. This is achieved with a public/private key pair. They operate as reverse functions to each other, which means data encrypted with the private key can be decrypted with the public key and vice versa. The assumption for the following considerations is that the server has its key pair already generated. This is usually done when setting up the LDAP server.

The simplified interchange between a client and a server negotiating an SSL/TLS connection is explained in the following steps:

1. As a first step, the client asks the server for an SSL/TLS session. The client also includes the SSL/TLS options it supports in the request.

2. The server sends back its SSL/TLS options and a certificate which includes, among other things, the server's public key, the identity for whom the certificate was issued (as a distinguished name), the certifier's name and the validity time. A certificate can be thought of as the electronic equivalent of a passport. It has to be issued by a general, trusted Certificate Authority (CA) which vouches that the public key really belongs to the entity mentioned in the certificate. The certificate is signed by the certifier which can be verified with the certifier's freely available public key.

3. The client then requests the server to prove its identity. This is to make sure that the certificate was not sent by someone else who intercepted it on a former occasion.

4. The server sends back a message including a message digest (similar to a check sum) which is encrypted with its private key. A message digest that is computed from the message content using a hash function has two features. It is extremely difficult to reverse, and it is nearly impossible to find a message that would produce the same digest. The client can decrypt the digest with the server's public key and then compare it with the digest it computes from the message. If both are equal, the server's identity is proved, and the authentication process is finished.

5. Next, server and client have to agree upon a secret (symmetric) key used for data encryption. Data encryption is done with a symmetric key algorithm because it is more efficient than the computing-intensive public key method. The client therefore generates a symmetric key, encrypts it with the server's public key, and sends it to the server. Only the server with its private key can decrypt the secret key.

6. The server decrypts the secret key and sends back a test message encrypted with the secret key to prove that the key has safely arrived. They can now start communicating using the symmetric key to encrypt the data.

As outlined above, SSL/TLS is used to authenticate a server to a client using its certificate and its private key and to negotiate a secret key later on used for data encryption.

3

Planning your directory

The first sections in this chapter describe some guidelines on how the design and implementation of the data and directory tree structure should be done. Then security planning is described, followed by implementing such a directory in a physical infrastructure having scalability, availability, manageability, and maintenance aspects of an LDAP directory deployment in mind.

Discussing low-level details of designing a directory implementation, such as detailed performance tuning aspects or product selection criteria, is beyond the scope of this book. However, this chapter gives you an introductory understanding of what has to be considered when LDAP is to be introduced in an organization.

The discussions that follow in this chapter often refer to typical White Pages directory implementations for people directories. This approach was chosen for the sake of simplicity. Please bear in mind, LDAP is not only suitable for people directories. An LDAP directory can hold almost any kind of information and can therefore be used for a much broader range of applications. The Directory-Enabled Networks Initiative (DEN) is just one example where an LDAP directory is being used for storing network configuration and topology data.

Creating a design that has the flexibility to accommodate changes within the organization is probably the single most important task in implementing a directory service. This will help save time and money as the directory service

57

grows. When designing the directory service, the project can be divided into several smaller projects:

► Surveying the directory service contents
► Creating access control strategies
► Replication and partitioning strategies
► Network planning (physical planning)

This chapter discusses the four main planning phases when designing an LDAP directory and briefly discusses implementation issues:

► The first phase, defining directory content, has two components. The first component, defining directory requirements, is about a careful analysis of the main purpose of the directory and the associated considerations to arrive at an overall approach to the directory plan. The second component, data design, is then about understanding the sources and nature of the data, deciding the scope of the data within the directory, and planning the way in which it will integrate with external data.

► The second phase, organizing your directory, also has two components. The first component, schema design, determines the format in which the data is to be stored. This is analogous to the field data definitions in a relational database. The second component, namespace design, determines the hierarchical structure of the directory. This is analogous to the relationship between individual files and their access paths in a relational database.

► The third phase, securing directory entries, is all about privacy and security design to ensure that the data in the directory is protected, as well as about allowing applications themselves to be secured by use of the directory. This aspect of the design affects all other aspects.

► The fourth phase, designing your server and network infrastructure, has two components. The first component, topology design, helps to determine the number and location of directory servers and how the data is distributed among them. The second, optional component, replication design, enables multiple copies of the data to be deployed, which can aid performance.

Surprising as it may seem, with the exception of security, the various major dimensions of design are largely independent of each other.

Some aspects of the design process allow for flexibility when requirements may change in the future. Others are less forgiving and can involve a major upheaval. It is essential to undergo a thorough planning process before starting the live implementation. Do not be misled into thinking, for instance, that because the directories' servers such as ITDS are included with various IBM Operating Systems, for example, included in the price of AIX®, it is a lightweight piece of infrastructure. Nothing could be further from the truth. In building an LDAP-enabled directory you are laying the framework for generations of software

that are even now beginning to emerge. The directory, like the database, is one of the major building blocks of your infrastructure and some attention to planning at the initial stages will reap rich rewards in the future.

We have discussed here some aspects of directory design. However, it needs to be pointed out that there is no single correct way to design a directory. To be able to build a more objective picture of the naming methodology, we recommend that several sources of information are compared. Often, vendors will have their own implementation guides that reflect different angles of views for this aspect.

3.1 Defining the directory content

The first phase, defining the directory content, is concerned with what it is that your proposed directory project sets out to achieve and what data is available to help it do so.

3.1.1 Defining directory requirements

This section discusses the directory definition requirements that need to be considered when planning a directory implementation.

Application needs

What type of application(s) will use the directory? Determine what directory-enabled applications are to be deployed and what are their data needs. Determine the organization's other mission-critical applications. Find out if those applications can directly access and/or update the directory. What are the requirements for manageability and scalability? Will the LDAP service be participating with an X.500 directory service?

User needs

Determine who needs access to the data as a user. Find out if those users can directly access or even update the directory. Determine the location of clients (users or applications). What expectations are there for privacy concerns? How accurate and up-to-date must the directory content be?

Deployment issues

What resources will be available for deployment? What people and skills are available? Can this be done as part of another project, for example, messaging migration, or will it require dedicated resources?

Infrastructure constraints

What hardware configurations are already in use and which, if any, are available to the project? What operating systems, middleware, and applications are in use? Specifically, what directory applications are already available? Obtain a network diagram. Is the directory to be protected behind a firewall or exposed to the Internet?

3.2 Data design

Planning the directory's data is the most important aspect of the directory planning activities, and it is probably the most time-consuming aspect as well. A

considerable amount of the time spent planning the directory data will most likely be spent surveying the organization to locate all the data stores where directory information is managed. As this survey is performed, expect to find that some kinds of data are not well-managed; some processes may be inefficient, inadequate, or non-existent; and some kinds of data may not be available at all. All of these issues should be addressed before finishing a data-planning phase.

However, we start by looking at the requirements on the data to be used in the directory service. The scope of information required will largely be driven by the application requirement. However, some types of data are better suited for a directory service than others. Ideal candidates in a directory service have some of the following characteristics:

► A directory service is not a file system, a file server, an FTP server, a Web server, or a relational database. Therefore, large, unstructured objects of data should not be put in the directory. For that kind of data, a server more appropriate for the task should be used. However, it is appropriate to store pointers to these kinds of applications within the directory service through the use of FTP, HTTP, or other types of accesses.

► The data should typically be read much more often than it is written. This is because directory services usually are tuned for read operations; write operations are more expensive in terms of resource utilization than reads, and they may impact the directory server's performance in typical directory server implementations.

► Another "rule of thumb" is that the data should typically be accessed from more than just one system or client. For example, an employee's preference settings for a specific application may not be meaningful to put in the directory if that application is only run on the employee's single workstation. If the user wants to run this application on different systems, such as a mail client application, then the application would certainly benefit from a central directory for storing user preferences. This would allow the employee to use the same setup on multiple systems or even platforms within the organization.

Having in mind the types of data suitable and unsuitable for use in a directory, it is now possible to survey what the directory service data will be.

3.2.1 Sources for data

Planning the directory content includes deciding which existing data to store in the directory. Survey the organization and identify where the data comes from (such as Windows NT® domains, RACF®, application-specific directories, human resources databases, e-mail systems, and so forth).

When deciding on what to put into the directory, all the owners of data relevant to the contents of the directory should be identified. It is very probable that the

information you will be choosing to put in the LDAP directory already resides on some other system in your organization. For example, the Personnel Department most likely already has databases with personnel information. Also be sure to make adequate use of processes already in place to administer that data even in the planned directory service.

Data management and access control are both important when maintaining a directory service. Plans must be made to identify resources for keeping the data up-to-date and identifying resources with the authority to decide on access control policies regarding the data residing in the directory tree.

If data is going to be imported from other sources, develop a strategy for both bulk imports and incremental updates. Try to limit the number of applications that can change the data. Doing this will help ensure the data integrity while reducing the organization's administration.

Identify duplications and data that is not actually used or required. Harmonize the data by eliminating such duplications and discard unnecessary data.

3.2.2 Characteristics of data elements

Data is made up of data elements, which possess several characteristics such as format, size, frequency, ownership, relationship with other data elements, etc. For instance, the data element e-mail address has a format of text, has many characters, has possible multiple values, is owned by the IT department, is used by both users and applications, and is related to the user's entry. Examine each planned data element to determine its characteristics and which are shared with other elements.

For each piece of data, determine the location where it will be mastered and who owns the data—that is, who is responsible for ensuring that the data is up-to-date.

3.2.3 Related data

Remember to plan for related data sources that contain directory-related data but which may not, initially at least, use the directory itself. For example, the human resources database must bear a close relationship to entries in a directory containing staff data. Consider appropriate replication and synchronization techniques and procedures to maintain the relationships.

3.3 Organizing your directory

Having decided on the type of data to use in the directory service, what the directory will be used for, and how the data will be updated, it is possible to start structuring the data. Structuring data is done by designing both a schema and a namespace. We explain these activities in the sections that follow.

3.3.1 Schema design

The schema design plays an important role in your directory implementation and helps you organize the data within a directory.

Directory schema

A schema is the collection of attribute-type definitions and object class definitions. A server uses these to determine how to match a filter or attribute against the attributes of a specific entry and whether to permit given attribute(s) to be added. This is similar to the data definitions of a relational database system. For more information on schemas, refer to "LDAP schema" on page 37.

Purpose

The purpose of a schema is to control the nature and format of the data stored in the directory. This means that schemas can be used for data validation and to control redundant data. A schema is also used by users and applications as the basis for directory search criteria.

Elements of LDAP schemas

LDAP directory schemas consist of attributes and object classes. A more detailed discussion on schema elements can be found in "LDAP schema" on page 37.

Design overview

Schema design involves several stages. First, identify any schemas provided by the applications you have in plan, plus any standard and vendor-supplied schemas. Secondly, select any predefined schemas that meet your needs. Thirdly, plan for any schema extensions.

For each piece of data, determine the name of the attribute(s) that you will use to represent the data in the directory and the object class(es) (the type of entry) that the data will be stored on.

Predefined schemas

When deciding on the design of the schema, there are a few things to consider. The LDAP specifications include a standard schema for a typical White Pages directory (RFC 2256, *A Summary of the X.500(96) User Schema for use with LDAPv3*). Vendors ship schemas with their LDAP server products that may include some extensions to support special features they feel are common and useful to their client applications. Work at the Internet Engineering Task Force (IETF) is in progress to create standard schemas for a broad range of applications.

Regardless of the type of information contained in the directory server, the standard schema, some of which is based on the X.500 standard, should not be modified. If this standard schema proves to be too limiting for the intended use, it can be extended to support the unique requirements. Standard schema elements, however, should not be deleted. Doing so can lead to inter-operability problems between different directory services and LDAP clients.

It is important to use a consistent schema within the directory server because LDAP-enabled application clients locate entries in the directory by searching for object classes or attributes and their associated values. If the schemas are inconsistent, then it becomes virtually impossible to locate information in the directory tree efficiently. An example of an inconsistent schema is a situation where an attribute is used to store a specific kind of information, and then later a different attribute is used to store the exact same kind of data, for example, when both attributes, telephoneNumber and phone, contain the same data.

Most LDAP-enabled application clients are designed to work with a specific, well-defined schema. Shrink-wrapped standard applications usually only work with a standard schema. These are important reasons why LDAP-based Directory Services should support at least the standard LDAP schema. Then the schema may be extended as the site discovers site-specific needs that are not met by the standard schema.

New schema elements

The use of a standard schema is beneficial, and specific changes can be made so long as they are additions. You may, however, create your own, private schema. But when doing so, you must take into consideration that compatibility to any other LDAP service may be lost and that your application clients have to be aware of that private schema.

3.3.2 Namespace design

Namespace design is a very important task in planning the directory. It is one of the most difficult to change at a later stage. A namespace is the means by which

directory data is uniquely named and referenced. It is the equivalent of the *unique key field* for the entry. The structure of an LDAP namespace is described in Chapter 2, "LDAP concepts and architecture" on page 27.

Purpose

The namespace provides a way to organize the data. It can be used to partition (group) the data and to provide a basis for replication. It can affect your access control methods. Finally, it is the basic support for directory-enabled applications.

Analyzing needs

Before designing your namespace you need to understand the requirements for it. Do you need a flat namespace or a hierarchical one? What attributes can be used to name entries? Do you anticipate replication or partitioning? Does a corporate taxonomy (hierarchical map of the organization) exist, and could or should it be used? Might your requirements change over time, for example, with company mergers and acquisitions?

Namespace design approach

Namespace design is done by choosing a directory suffix, branching the directory tree, and finally creating a naming style for the directory entries.

Choosing a suffix

When deciding on suffixes, where a suffix is the root DN of a directory tree, it is a good idea to use the same naming structure for LDAP as is used for X.500. Using the X.500 methodology would lead to choosing a suffix like o=ibm,c=us or ou=raleigh,o=ibm.

This method will set the root of the directory tree to a specific organization in a specific country or to a specific organization and organizational unit. However, it is not necessary to do this, unless there are plans to participate in an X.500 directory service, since LDAP does not require any specific format for the DN naming convention. In LDAP, the directory suffix can be chosen freely to reflect the organizations distinct name. Another method that you can use, if the X.500 method does not seem appropriate, is the DNS naming model when choosing the directory suffix. This would result in a suffix using the domainComponent attribute, for example, dc=server,dc=company,dc=com.

The design of the directory schema and definition of the suffix makes it possible to start populating the tree. But, before doing so, the naming structure must be put in place. We have divided the discussion on naming structure creation into the two sections that follow:

► Branching of the directory tree
► Naming style for the entries

Branching the directory tree

Choosing to branch a directory tree based on the organizational structure, such as departments, can lead to a large administrative overhead if the organization is very dynamic and changes often. On the other hand, branching the tree based on geography may restrict the ability to reflect information about the organizational structure. A branching methodology that is flexible, and which still reflects enough information about the organization, must be created.

Because the structure of organizations often changes considerably over time, the aim should be to branch the tree in such a way as to minimize the number of necessary changes to the directory tree once the organization has changed. Note that renaming a department entry, for example, has the effect of requiring a change of the DNs of all entries below its branch point. This has an undesirable impact on the service for several reasons. Alias entries and certain attributes or ordinary entries, such as seeAlso and secretary, use DNs to maintain links with other entries. These references are one-way only, and LDAP currently offers no support to automatically update all references to an entry once its DN changes. The impact of renaming branches is illustrated in the following example.

When adding employees to their respective departments, it would be possible to create distinguished names (DN) like cn=John Smith, ou=Marketing, l=se, and dc=xyz.com. If John Smith should at a later time move to another department, his DN will have to change. This results in changing all entries regarding access rights and more. If John Smith's DN had been set to cn=John Smith, ou=employees, l=se, dc=xyz.com, then this would not be a problem. An attribute describing which department he belongs to (ou=marketing) could be added to his entry to include this information.

Other criteria that may or should be considered when branching the directory tree include physical or cultural splits in the organization and the nature of the client (human or application).

If your organization has separate units that are either physically separated or have their own management authorities, you might have a natural requirement to split and separate parts of the DIT.

A general rule of thumb says that the DIT should be reasonably shallow unless there are strong reasons to design deep branching levels down the directory tree. If the directory information is primarily searched and read by human users (that is, if users manually type in search criteria), the DIT should provide the information in an intuitive manner so that finding information is not limited to system specialists. If, on the other hand, the information is primarily retrieved from programs, other rules more suitable for that application can be followed.

3.3.3 Naming style

The first goal of naming is to provide unique identifiers for entries. Once this is achieved, the next major goal should be to make querying of the directory tree intuitive. Support for a naming structure that enables the use of user-friendly naming is desirable. Other considerations, such as accurately reflecting the organizational structure of an organization, should be disregarded if it has a negative effect of creating complex DNs, thus making normal querying non-intuitive. If we take a look at the X.500 view on naming, we see that the X.501 standard specifies that "RDNs are intended to be long-lived so that the users of the Directory can store the distinguished names of objects...", and "it is preferable that distinguished names of objects which humans have to deal with be user-friendly" (excerpt from *The Directory - Overview of Concepts, Models and Services*, CCITT 1988, cited in RFC 1617).

Multicomponent relative distinguished names can be created by using more than one component selected from the set of the attributes of the entry to be named. This is useful when there are, for example, two persons named John Smith in one department. The use of multicomponent relative distinguished names allows one to avoid artificial naming values such as cn=John Smith 1 or cn=John Smith 2. Attributes that could be used as the additional naming attribute include title, room number, telephone number, and user ID, resulting in a RDN, like title=Dr, cn=John Smith, creating a more user-friendly naming model.

A consistent approach to naming people is especially important when the directory stores information about people. Client applications will also be better able to assist users if entries have names conforming to a common format, or at least to a very limited set of formats. It is practical if the RDN follows such a format.

In general, the standard attribute types should be used as documented in the standards whenever possible. It is important to decide, within the organization, which attributes to use for what purpose and not to deviate from that structure.

It is also important that the choice of a naming strategy not be made on the basis of the possibilities of the currently available client applications. For example, it is questionable to use commonName of the form *surname firstname* merely because a client application presents results in a more satisfactory order by doing so. Use the best structure for people's names, and adapt or design the client applications accordingly.

Please refer to "LDAP distinguished name syntax (DNs)" on page 43 for a more detailed explanation of LDAP Distinguished name syntax.

3.4 Securing directory entries

Having designed the directory tree, we now need to decide on a security policy.

The degree of security controls you require will depend on the nature of the information you are storing. If it is just e-mail addresses then the worst danger of unlimited read capability is spam e-mail, and the worst danger of uncontrolled editing is misdirected e-mail. However, if the directory contains gender, home addresses, and social security numbers then the dangers are more extensive.

The degree of security you require will also reflect the ways in which clients will be accessing the directory and the methods that will be used to update and manage the directory.

Finally, it needs to reflect an acceptable level of administration effort for security. A security policy should be strong enough to prevent sensitive information from being modified or retrieved by unauthorized users, while simple enough that administration is kept simple so authorized parties can easily access it. Ease of administration is very important when it comes to designing a security policy. Too complex a security policy can lead to mistakes that either prevent people from accessing information that they should have access to, or allow people to modify or retrieve directory information that they should not have access to.

3.4.1 Purpose

The most basic purpose of security is to protect the data in your directory. It needs to be protected against unauthorized access, tampering with information, and denial of service.

3.4.2 Analysis of security requirements

Try to find answers to the following sorts of questions. Will your directory be read-only? How sensitive is the data? Is replication to multiple locations planned? What privileges might administrators have? How reliable are the users? How will they react to different levels of security? Will they require access across the Internet? Is your network itself secure? How about the machine room?

3.4.3 Design overview

To plan for the required level of security, two basic areas must be considered to answer the following questions: What level of security is needed when clients identify themselves to the directory server, and what methodology will be used

when authorizing access to the different kinds of information in the directory? These areas are authentication and authorization.

3.4.4 Authentication design

Conceptually, directory authentication can be thought of as logging into the directory. LDAP terminology, however, usually refers to this operation as binding to the directory.

Generally, bind operations consist of providing the equivalent of a user ID and a password. However, in the case of an LDAP directory, the user ID is actually a distinguished name (or a distinguished name derived from a user ID). The distinguished name used to access the directory is referred to as the bind DN.

So, what level of authentication should be considered? There are, generally speaking, three different approaches:

► *No authentication:* This is the simplest approach, which might be perfectly suitable for most directories when all users are equally granted read (or even write) access to all data. There is no need for user authentication when this is the case.

► *Basic authentication:* This lets the client bind by entering a DN and a password. Using basic authentication will not ensure integrity and confidentiality of the login data since it is being sent over the network in a readable form.

► *Secure authentication:* Simple Authentication and Security Layer (SASL) is an extensible authentication framework. It was added to LDAP Version 3, and it supports Kerberos and other security methods, like S/Key. SASL provides the ability to securely authenticate LDAP clients and LDAP directory servers. There is an external mechanism in SASL that allows the use of authentication identity information from security layers external to the SASL layer. One possibility is to use the authentication information from SSL. SSL is generally used to secure the connection between a client and a server through the exchange of certificates. The client certificate can be used through SASL as authentication identity. SASL is already used within several Internet protocols including IMAP4 and POP3 (mail server protocols).

It is possible that there is a need for both basic and secure authentication. The choice will be dependent on the security policies in the organization's networks and what type of access rights the different types of clients will have when communicating with the server. For example, when setting up server-to-server communication, it may be valuable to use strong, secure authentication since server-to-server communication will often rely on unrestricted access to each other's tree structures, including individual entry's access settings. On the other hand, for client-to-server communication, where clients only have read access to

names, phone numbers, and mail addresses, there is most likely no need for anything but basic authentication.

When using secure authentication, it is possible to choose from different methods depending on the vendors' implementations, for example, Kerberos or SSL. If Kerberos is not already deployed in the organization's intranet, then it will probably be sensible to use SSL, since support for SSL is included in most popular LDAP clients. When using SSL, it is possible for the server to authenticate the client by using its server certificate. A server certificate can be thought of as a secure, digital signature that uniquely identifies a server. It has been generated and registered with a trusted certifying authority, also known as a Certificate Authority (CA), such as VeriSign or the IBM World Registry™ CA. Also, when using server certificates, an encrypted communication can be established between the client and server, enabling a secure basic authentication of the client to the server.

Using SSL server certificates will be particularly interesting when setting up LDAP services on insecure networks, such as the Internet/extranet. This will enable the clients to verify the identity of the server and to encrypt communication of the basic authentication from the clients to the server on the insecure networks.

When using basic authentication, administration of passwords on the directory server will be necessary and may impose some administration overhead. If SSL client certificates are used, then an appropriate infrastructure will be needed to support the certificate generation and administration. This is usually done by separate certificate servers. Client certificate deployment is beyond the scope of this book, but it ought to be mentioned that LDAP supports storing client public keys and certificates in the entries allowing you also to use the directory by mail clients to encrypt e-mail.

3.4.5 Authorization design

The data in the directory tree will have to be protected in different ways. Certain information must be searchable for everybody, some must be readable, and most of it will be write protected. In LDAP Version 3, there are no defined attributes to handle this. As a result, vendors support their own implementations of authorization. This is done by different implementations of access control lists (ACLs).

ACLs are used to define access rules to the different entries in the directory tree. As an example of an ACL implementation, Example 3-1 on page 71 shows the IBM ITDS implementation of ACL attribute entries. The pertinent control attributes used here are aclsource, aclpropagate, and aclentry, where the latter, for example, is the attribute that specifies who has access to the entry and what

level of access he or she has. In Example 3-1, cn=John
Arnold,ou=employees,o=iseriesshop has read, write, search, and compare
(rwsc) rights for normal, sensitive and critical data (the entry is highlighted and
split into three lines in the example).

Example 3-1   Sample ACL attribute entry

```
dn: ou=employees, o=ibm, c=us
objectclass: top
objectclass: organizationalUnit
ou: employees description: Employees of IBM Corporation
entryowner: access-id:cn=admin,o=ibm, c=us
inheritoncreate: TRUE
ownerpropagate: TRUE
aclpropagate: TRUE
ownersource: default
aclsource: OU=employees,o=ibm, c=us
aclentry: access-id:CN=John
Arnold,OU=employees,o=ibm,c=us:object:a:normal:rwsc:
sensitive:rwsc:critical:rwsc aclentry: group:CN=ANYBODY:normal:rsc
```

When setting up access control lists, it is important to do it with the goal of
minimizing the administration later on. It is good to try and delegate the access
control hierarchically. An example of this could be the following: An individual,
say John Arnold, needs to protect sensitive information. Two groups have been
created for this purpose, owned by John Arnold (shown in Table 3-1). Entries can
be added and deleted by John Arnold to his own groups without intervention of
the directory service administrators.

Table 3-1   ACL structure for Web content administration using two groups

Group name	Owner	Group members
cn=editor	cn=Debbie Smith	cn=user1 cn=user2
cn=readers	cn=Brian Arnold	cn=user3 ou=techsupport

According to the table, John Arnold has added `user1` and `user2` to the `editor`
group and `user3` and the group called `techsupport` to the `readers` group, thus
enabling `user1` and `user2` to edit the contents, and enabling `user3` and the
people in the `techsupport` organizational unit to read the contents.

3.4.6 Non-directory security considerations

Other security considerations that are not directly related to directory design but
that can help to protect your data include encryption.

You should also ensure that your organization's security audit procedures are updated to reflect the new directory plan.

3.5 Designing your server and network infrastructure

Physical design involves building a network and server infrastructure to support availability, scalability, and manageability. Methods to do this in LDAP are partitioning and replication. Replication is actually not standardized in LDAP Version 3, but all directory vendors implement replication within their products. In this section we concentrate on deployment issues regarding when partitioning and/or replication is appropriate when trying to reach the goals of availability, scalability, and manageability, and what the trade-offs are.

In sizing the directory service, consideration must be given to which clients will be accessing what data, from where, and how often. If there are client applications that use the directory extensively, consideration must be given to ensuring that the network availability and bandwidth are sufficient between the application servers and the directory servers. If there are network bottlenecks, they must be identified because there may be a need to replicate data into remote LANs.

3.5.1 Availability, scalability, and manageability requirements

Availability for a directory service may not be a hot issue in cases where the directory is not business critical. However, if the use of the service becomes mission critical, then there is a need to design a highly available system. Designing a highly available system involves more than is supported in LDAP. The components from LDAP that are needed are partitioning and replication. Since high availability involves eliminating single points of failure or reducing their impact, it is necessary to have redundant hardware, software, and networks to spread the risk.

As more and more applications use and rely on a directory service, the need to scale the directory for high-load tolerance increases. Scaling up directory servers is done much the same way, either by increasing availability or by upgrading hardware performance. As is the case when increasing availability, we have to rely on functions outside the LDAP standard as well as LDAP replication and partitioning. The round-robin DNS or the load-balancing router, such as the IBM WebSphere Edge Server, are good tools to scale an LDAP server site.

Manageability aspects involve almost all parts of a directory design. Here is where trade-offs may have to be made regarding scalability, availability, flexibility, and manageability. The level of scalability and availability are both related to cost in hardware and software and, as a drag-along, cost of overall

systems management. One important question to ask in a directory design about manageability is whether and how all information providers are able to furnish reliable, correct, and consistent directory data to the LDAP service. If this cannot be assured, there will be a chance for errors and inconsistencies in the LDAP directory data. If such problems are considered critical for the clients using the LDAP service, tools must be provided that can detect and maybe even correct these errors.

3.5.2 Topology design

Topology design concerns the distribution of directory servers. The first choice is between a centralized or a distributed approach. The second choice is between a partitioned and a replicated approach.

Centralized or distributed

You can choose to centralize in a single master directory or to distribute the data to additional directory servers.

A simple approach to create a highly available directory service is to create a master and a replica directory server, each one on its own physical machine. By replicating the data, we have eliminated the single point-of-failure for both hardware and software failures. This solution with a master and one or more replica servers normally provides for high availability for read functions to the LDAP servers. Write requests can only be directed to the master server. If high availability is required for write access, additional effort is necessary. Neither read-only nor read/write replication is supported natively by the LDAP standards, but vendors may have implemented their own mechanisms. Replication solutions can also be constructed using the export/import facilities of LDAP servers or with additional, custom-designed software tools. Also the OS/400® Directory server has its own replication mechanism that is constantly being enhanced.

A mechanism must be added to handle client redirection if one server fails. This can be done manually or semi-automatically by a DNS switchover, or automatically with a load-balancing technique by using a router designed for this. Such a router forwards client requests to one of the servers based on configurable criteria. It is important that the router supports stateful protocols; that is, subsequent requests from the same client need to be forwarded to the same server. There are several products on the market from different vendors to do this, such as IBM's WebSphere Edge Server. This function is also built into IBM Lotus Domino. The IBM @server iSeries of course allows multiple Domino server instances to run within a single operating system instance.

There is also the issue of network bandwidth and its reliability to take into consideration. In some cases, it may be necessary to distribute a replica into another LAN with slow network connections to the master. This can also be done

with any means of replicating an LDAP server (remember that replication is not included in the LDAP standards, thus you have to use vendor product support or your own methods). The primary server for a particular client may be the directory server on the client's own LAN, and the secondary will then be the central master server, accessed over the WAN.

If the method of spreading the risk is used to create high availability, it is possible to partition the directory tree and to distribute it to different locations, LANs, or departments. As a side-effect, depending on how the directory tree is branched and distributed to these servers, each location, department, or LAN administrator could then easily manage their own part of the directory tree on a local machine, if this is a requirement. If a single server failed in such a configuration, then only a portion of the whole directory would be affected.

A combination of these methods could be used to create a dynamic, distributed, highly available directory service.

Partitioned or replicated

The second choice for topology design is only applicable when a distributed approach has been selected for the first choice. The options are between a partitioned and a replicated approach. The decision criteria are usually based on performance and availability issues and will be influenced by the size of the directory.

To create a high-availability environment, it is necessary to replicate and/or partition the directory, as discussed in the previous sections. Although not directly related to LDAP, it should be mentioned that adequate systems management tools and skills must be available to run such a fairly complex environment. In addition, one of the manageability concerns regarding replication might be the need to ensure an ample level of consistency. A master LDAP server might have been updated with new information, while a replica server still runs with the old, outdated information. The required level of consistency is largely dependent on the needs of the client applications using the service. If there is a requirement for currency and consistency among replicated servers, additional means must be provided to ensure this.

Replication will also affect back up and disaster/recovery procedures. Processes will be needed to handle recovery of master servers and how synchronization of replicas will be handled. Since replication is outside the current standard for LDAP, it is necessary to study the vendor's implementation in order to find adequate solutions.

Partitioning the directory enables local servers to own their own data, depending on schema and branching design. This increases flexibility when maintaining data, but increases the complexity of referral handling. A clear method of linking

the name space together will have to be formulated to ensure consistent referrals in the directory service name space such that the logical name space is still a whole. Also, each local server may have to be administered and maintained locally, requiring staff with operating system and LDAP knowledge.

You should consider partitioning if the directory is very large, if your applications only require local workgroup data, if replication volumes would otherwise be too big, if your WAN is not suited to high volumes, and where future expansion of the service might trigger one of these considerations in the future.

The optimal topology design depends on the applications, the server, the physical network, and the directory namespace.

Remember that each partition needs a partition root, which is the DN of the entry at the top of the naming context, and hence occurs at a branching point in your directory. You may need to revisit your namespace design.

3.5.3 Replication design

Replication is a technique used by directory servers to improve performance, availability, and reliability. The replication process keeps the data in multiple directory servers synchronized.

Replication provides three main benefits:

► Redundancy of information: Replicas back up the content of their supplier servers.

► Faster searches: Search requests can be spread among several different servers, instead of a single server. This improves the response time for the request completion.

► Security and content filtering: Replicas can contain subsets of the data in a supplier server.

The replication design stage is only required when, firstly, a distributed approach is chosen to server deployment and, secondly, a replicated approach is chosen over a partitioned approach. Replication aims to improve the reliability and performance of your directory service.

Concepts

By making directory data available in more than one location you improve the reliability of the service in the event of server or network failure. You also improve the performance by distributing the load across multiple servers and reducing network traffic.

Designing replication

Consider first the unit of replication. This concerns which entries and which of their attributes are to be replicated. A subtree of the DIT might form a suitable selection basis. Now think about how consistent the data has to be. Must every change be replicated instantly to all servers? Due to the nature of directory data, for example people's phone numbers, it is not usual to impose such a tight restriction, but you might take a different view of removing the entry for a dismissed member of staff. Think about the sort of replication schedules that might be appropriate for your directory and network. Also, if you replicate Certificate Revocation Lists (CRLs) you may want to replicate information about a revoked certificate instantly.

To ensure initial copies are in place we might use LDAP Data Interchange Format (LDIF) files to import volumes of data in batch. A more incremental approach might be used for subsequent updates.

What sort of replication strategy is appropriate? Is a master-replica approach suitable, with all changes being driven out from the center? The alternative is a peer-to-peer approach, which allows all servers to update their own data and subsequently to exchange it.

The replication capabilities of various vendor's LDAPv3 directory servers widely vary. It is advisable to look at your particular vendors documentation on replication to understand what features and capabilities exist in each respective product. Nearly all of the existing directory server implementations support the following three types of replication topologies in one form or another.

Master-Replica Replication

The basic relationship in replication is that of a master server and its replica server. The master server can contain a directory or a subtree of a directory. The master is writable, which means it can receive updates from clients for a given subtree. The replica server contains a copy of the directory or a copy of part of the directory of the master server. The replica is read only; it cannot be directly updated by clients. Instead it refers client requests to the master server, which performs the updates and then replicates them to the replica server.

The most simple example of the Master-Replica topology can be seen in Figure 3-1 on page 77. The Master Server in this example is replicating all of its data to the Replica server.
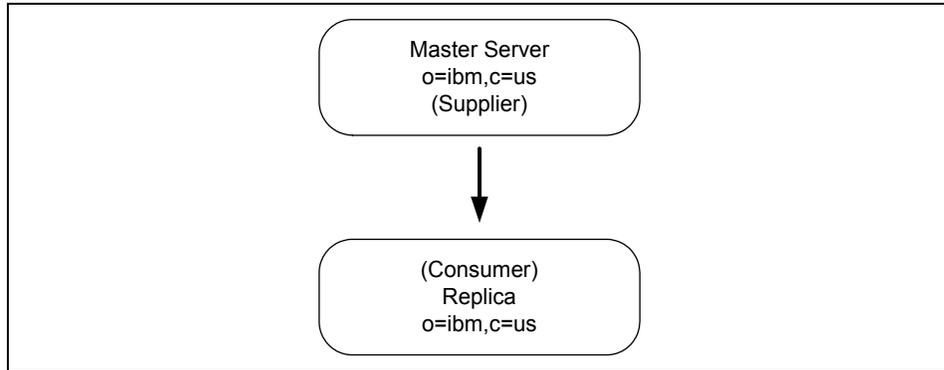
Figure 3-1   Master-replica replication topology (single consumer)

A master server can have several replicas. Each replica can contain a copy of the master's entire directory, or a subtree of the directory. In Figure 3-2, `Replica 2` contains a copy of the complete directory of the Master Server; `Replica 1` and `Replica 3` each contain a copy of a subtree of the Master Server's directory.
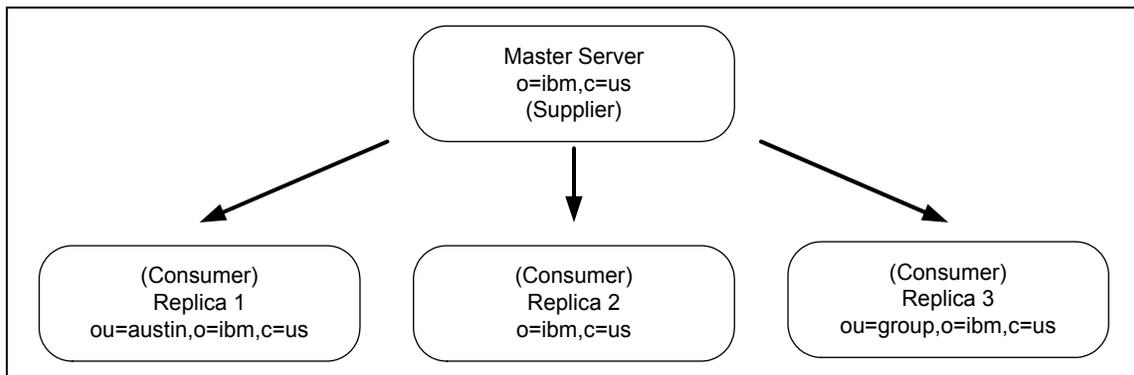
Figure 3-2   Master-replica replication topology (multiple consumers)

The relationship between two servers can also be described in terms of roles, either supplier or consumer. In Figure 3-2, the Master Server is a supplier to each of the replicas. Each replica in turn is a consumer of the Master Server.

Cascading replication

Cascading replication is a topology that has multiple tiers of servers. A master server replicates to a set of read-only (forwarding) servers that in turn replicate to other servers. Some vendors call these forwarding servers *replication hubs*. Such a topology off-loads replication work from the master server. In the example of this type of topology, the master server is a supplier to the two forwarding servers. The forwarding servers serve two roles. They are consumers

of the master server and suppliers to the replica servers associated with them. The replica servers are consumers of their respective forwarding servers. This is shown in Figure 3-3.
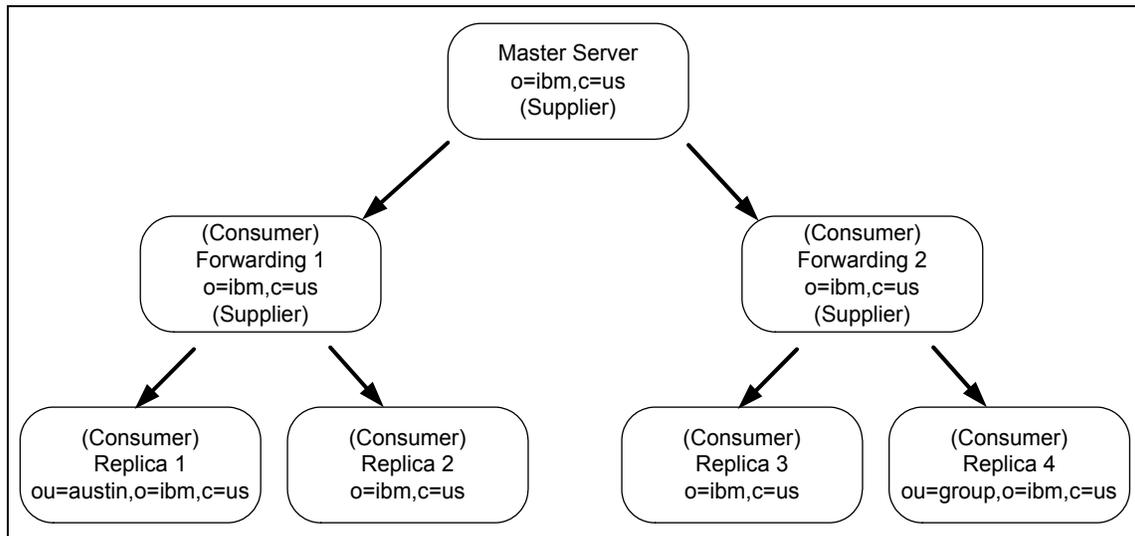
Figure 3-3   Cascading replication topology

Peer-to-peer replication

There can be several servers acting as masters for directory information, with each master responsible for updating other master servers and replica servers. This is referred to as peer replication. Some vendors also refer to this replication topology as *multi-master*. Peer replication can improve performance, availability, and reliability. Performance is improved by providing a local server to handle updates in a widely distributed network. Availability and reliability are improved by providing a backup master server ready to take over immediately if the primary master fails. Peer master servers replicate all client updates to the replicas and to the other peer masters, but do not replicate updates received from other master servers. Peer replication is shown in Figure 3-4 on page 79.
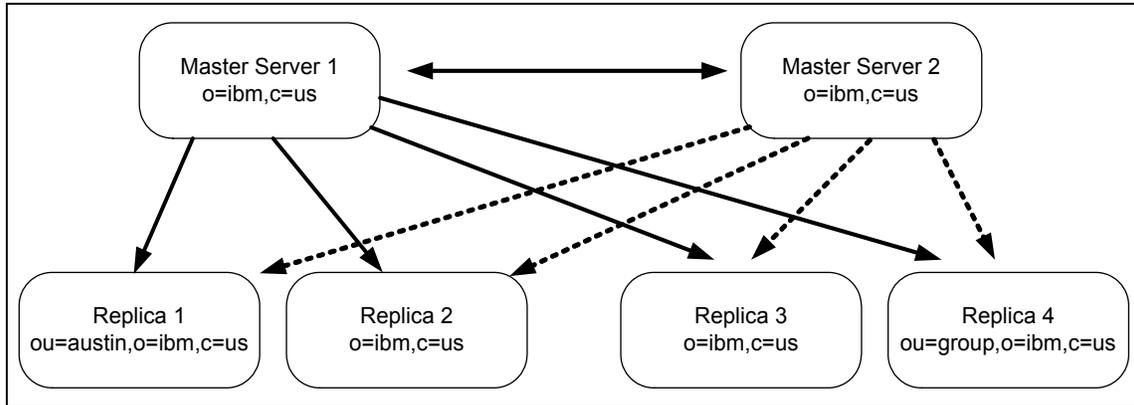
Figure 3-4   Peer-to-peer replication topology

3.5.4 Administration

In this section we show the tools for administering the directory, then we present a brief review of who should perform administrative tasks.

The LDAP specifications contained in the pertinent RFCs include functions for directory data management. These include functions to create and modify the Directory Information Tree (DIT) and to add, modify, and delete data stored in the directory.

Vendor products, however, most likely include additional tools for configuring and managing an LDAP server environment. These include such functions as:

► Server setup (initial creation)
► Configuring a Directory Information Tree
► Content management
► Security setup
► Replication and referrals management
► Access control management
► Logging and log file management
► Resource management and performance analysis tools

Depending on specific needs and preferences, LDAP directory administration can be performed several ways. Different vendors offer different administration tools. Although not all vendors provide tools for all methods, in general there are three tools to manage LDAP directories:

► Graphical administration tools
► Command line utilities
► Custom-written applications

Graphical tools features are specific to each vendor, when provided.

Command line tools are based on the LDAP Software Development Kit (SDK), which is mainly a set of libraries and header files. Depending on vendors, most SDKs come with a set of simple command line applications, either in source code or as ready-to-use executable programs. These tools were built using the LDAP API functions and thus can serve as sample applications. They enable you to do basic operations, such as searching the directory and adding, modifying, or deleting entries within the LDAP server. Each basic operation is accomplished with a single program such as *ldapsearch* or *ldapmodify*. By combining these tools using, for example a scripting language such as Perl, you can easily build up more complex applications. In addition, they are easily deployable in Web-based CGI programs.

As an alternative to using the administration utilities, custom-written administration tools can be used. A developer has several options for accessing LDAP. An API library for both C and Java languages is available. Another approach for custom-written tools is to use the Java Naming and Directory Interface (JNDI) client APIs. Such administration tools might be desirable when typical data administration, such as adding or modifying employee data, is done by non-technical staff. Writing directly to the API layer may also be necessary for applications that need to control the bind/unbind sequence, or, perhaps, want to customize the referral behavior. This is a more difficult approach because the developer must deal with the conversion of the data to the structures that are sent over the LDAP protocol. Additionally, the developer must be aware of a particular security setup, such as SSL.

Part 2

IBM Tivoli Directory Server overview and installation

In this part we provide an introduction to IBM's directory server offering, named IBM Tivoli Directory Server. We provide an overview of ITDS, cover installation and basic configurations for Mircrosoft Windows, IBM AIX, Intel Linux, and IBM zSeries operating systems.

IBM Tivoli Directory Server overview

This chapter provides an overview of IBM Tivoli Directory Server (ITDS) and provides a roadmap to the rest of the book, which focuses primarily on the installation, configuration, and operation of TDS.

4.1 Definition of ITDS

The IBM Tivoli Directory Server implements the Internet Engineering Task Force (IETF) LDAP V3 specifications. It also includes enhancements added by IBM in functional and performance areas. This version uses IBM DB2 Universal Database as the backing store to provide per-LDAP operation transaction integrity, high performance operations, and online backup and restore capability. The IBM Tivoli Directory Server interoperates with the IETF LDAP V3 based clients.

Figure 4-1 provides a high-level overview of what the various components of ITDS are and how clients might interact with it.
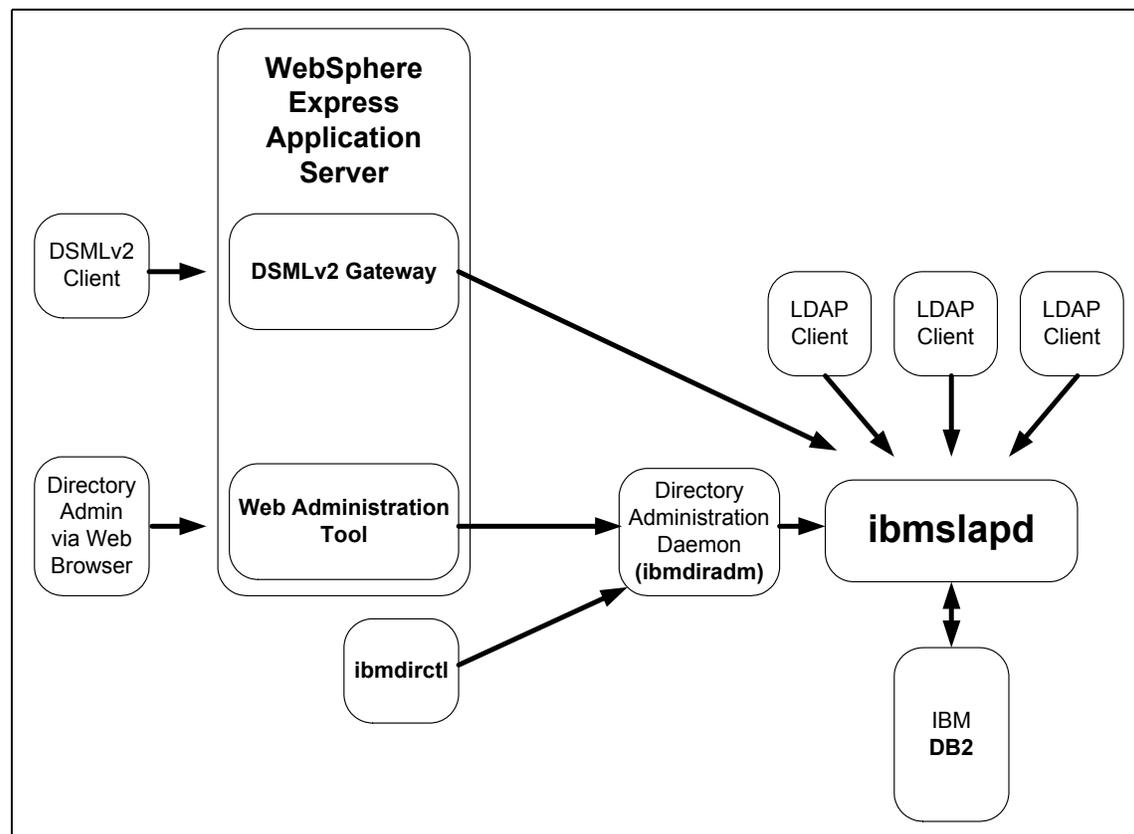
Figure 4-1   ITDS high-level overview

The base components of IBM Tivoli Directory Server are:

► IBM DB2 Universal Database as the backing store to provide per-LDAP operation transaction integrity, high-performance operations, and online

backup and restore capability. IBM Tivoli Directory Server Version 5.2 currently ships with DB2 V8.1.

► The server executable named *ibmslapd*.

► Tools to administer and configure the directory. These tools rely on the directory administration daemon (*ibmdiradm*), which runs on each server machine and also enables remote management. The main tools are:

  – Web Administration Tool. This is a J2EE compliance application installable on IBM WebSphere Application Server and in its Express version, which is provided with IBM Tivoli Directory Server—GUI for configuring the directory and the database: Configuration tool (*ldapxcfg*).

  – Command line server utilities.

  – IBM Tivoli Directory Server Client SDK, which provides the tools required to develop LDAP applications. It includes:

    • Client libraries that provide a set of C-language APIs

    • C header files for building and compiling LDAP applications

    • Documentation that describes the programming interface and the sample programs

    • Sample programs in source form

    • Command line client utilities

► DSMLv2 Front-end which provides DSMLv2 services via an application that run from the bundled IBM WebSphere Express server.

The major features of ITDS include:

► A Graphical User Interface (GUI) that can be used to administer and configure the IBM Directory. The administration and configuration functions enable the administrator to:

  – Perform the initial setup of the directory.

  – Change configuration parameters and options.

  – Manage the daily operations of the directory, such as adding or editing objects, for example object classes, attributes, and entries.

► A dynamically extensible directory schema. This means that administrators can define new attributes and object classes to enhance the directory schema. Changes can be made to the directory schema, too, which are subject to consistency checks. Users may dynamically modify the schema content without restarting the directory server. Because the schema itself is part of the directory, schema update operations are done through standard

LDAP APIs. The major functions provided by the LDAPv3 dynamic extensible schema are:

– Queriable schema information through LDAP APIs
– Dynamic schema changes through LDAP APIs
– Server Root DSE

► UTF-8 (Universal Character Set Transformation Format). An IBM Tivoli Directory Server supports data in multiple languages, and allows users to store, retrieve and manage information in a native language code page.

► Simple Authentication and Security Layer (SASL). This support provides for additional authentication mechanisms. The Secure Sockets Layer (SSL) provides encryption of data and authentication using X.509v3 public-key certificates. A server may be configured to run with or without SSL support.

► Replication. Replication is supported, which makes additional read-only copies of the directory available, improving performance and reliability of the directory service. Replication topologies also support forwarding and gateway servers.

► Referrals. Support for LDAP referrals, allowing directories to be distributed across multiple LDAP servers where a single server may contain only a subset of the whole directory data.

► Access control model. A powerful, easy-to-manage access control model is supported through ACLs.

► Change log.

► Password policy.

► Security audit logging.

► Dynamic configuration changes using LDAP APIs.

IBM Tivoli Directory Server is a powerful, security-rich and standards-compliant enterprise directory for corporate intranets and the Internet. Directory Server is built to serve as the identity data foundation for rapid development and deployment of your Web applications and security and identity management initiatives by including strong management, replication and security features.

With IBM Tivoli Directory Server you can choose your authentication strategy, you can use simple user ID and password authentication, or you can implement the more secure digital certificate-based authentication structure. IBM Tivoli Directory Server also includes a Simple Authentication Security Layer (SASL) plug-in interface, including Challenge-Response Authentication Mechanism MD5 (CRAM-MD5) and Kerberos authentication if required.

The fine grained access control features in IBM Tivoli Directory Server extend to the attribute level, enabling self service and delegated administration while also

offering protection of access control list (ACL) values within the directory, preventing unauthorized users from changing the security assigned to objects within the directory.

Development and deployment of your enterprise directory with IBM Tivoli Directory Server is enhanced through the inclusion of the IBM default schema, a flexible server plug-in framework and the client SDK which includes support for 64-bit AIX and Java TM access via a standard J2EE interface.

IBM Tivoli Directory Server is a component of the IBM Tivoli Identity Manager solution that can help you get users, systems and applications online and productive fast, reduce costs and maximize return on investment. IBM Tivoli Identity Manager provides identity lifecycle management (user self-care, enrollment and provisioning), identity control (access and privacy control, single sign-on and auditing), identity federation (sharing user authentication and attribute information between trusted Web services applications) and identity foundation (directory and workflow) to effectively manage internal users as well increase number of customers and partners through the Internet.

4.2 ITDS 5.2

ITDS, released in October 2003, introduced a number of new features well as enhancements to existing capabilities. These features and enhancements include:

► Updated versions of corequisite products

  – DB2 Universal Database Version 8.1 Enterprise Server Edition (DB2) with FixPak 2.

  – IBM Global Security Kit (GSKit) Version 7a. GSKit includes open-source libraries.

  – The embedded version of IBM WebSphere Application Server - Express Version 5.0.2.

► Support for Microsoft Windows Server 2003

  IBM Tivoli Directory Server supports the Microsoft Windows Server 2003 operating system, Standard and Enterprise editions.

► Non-SSL packages only on AIX

  In previous versions, both Secure Sockets Layer (SSL) and non-SSL packages were provided on all operating system platforms. For IBM Tivoli Directory Server Version 5.2, non-SSL packages are provided only on AIX.

- ► Full 64-bit server support on AIX

  IBM Tivoli Directory Server has been ported to 64-bit architecture on AIX only. Solaris, HP-UX, Linux zSeries, Linux Intel, Linux iSeries and pSeries®, and Microsoft Windows remain 32-bit servers. The Web Administration Tool remains a 32-bit application. The 32-bit server will no longer be available on AIX; however, the client SDK will still be available as a 32-bit application. The 64-bit architecture increases the ability to cache a large number of directory entries.

  Note that AIX Version 5.1 or later is required for the 64-bit AIX Server.

  To move up to 64-bit server support, you must migrate your database. However, you do not need to unload and reload your data. See the chapter on "Migration from previous releases" located in *Installation and Configuration Guide*, SC32-1338.

- ► *Authentication methods for LDAP* (RFC 2829)

  IBM Tivoli Directory Server 5.2 provides support for DIGEST-MD5 Simple Authentication and Security Layer (SASL) authentication, as well as Transport Layer Security (TL\S) support as defined in RFC 2829.

- ► *LDAP v3 Extensions for TLS* (RFC 2830)

  TLS allows clients to connect to the server on a non-secure port and issue a TLS start command. If GSKit is installed, the server honors the request and begins a secure connection with the client. RFC 2830 specifies how LDAP should support TLS.

- ► *DIGEST-MD5 SASL Mechanism* (RFC 2831)

  RFC 2831 defines how HTTP Digest Authentication (Digest) can be used as an SASL mechanism for any protocol that has an SASL profile. (RFC 2222 defines SASL.) DIGEST-MD5 is intended to be both an improvement over CRAM-MD5 and a convenient way to support a single authentication mechanism for Web, mail, LDAP, and other protocols.

- ► *Use of Language codes* (RFC 2596)

  RFC 2596 defines a mechanism that allows the directory to associate natural language codes with values that meet certain natural language requirements. IBM Tivoli Directory Server 5.2 supports a single language code option and language tag support discovery.

- ► Subtree search on null base

  A subtree can now be searched from a null base. This provides a shorthand way to retrieve all entries in the directory. In earlier releases, multiple searches were required for each suffix to search the entire directory.

► Unique attributes

IBM Tivoli Directory Server 5.2 allows the administrator to identify attributes that must have unique values. This ensures that there are not two directory entries with the same attribute values. For example, no two users can have the same user ID or e-mail address if these attributes have been configured to enforce uniqueness.

► Delegation of server administration to a group of administrative users

In previous releases, IBM Tivoli Directory Server required that the administrator user ID be used to perform server tasks such as replication configuration and starting and stopping the server. For the 5.2 release, there is an administration group that contains IDs of users with administrative rights and privileges. This avoids the use of a single administration ID shared by a number of administrators. The root administrator can add or remove members from the administration group.

► Prevention of denial of service

For the 5.2 release, support has been added to reduce the vulnerability of the server to malicious attacks, causing a denial of service. The server can be configured to reject non-responsive clients after some number of attempts. Support has also been added to close connections issued by a specific IP address or DN. An emergency thread is available when some number of items, configurable on the server, are on the work queue. This provides a method for the administrator to access the server during a denial of service attack. The oldest connections can, through configurable parameters, be reused first.

► Unbind of bound DN/IP

This security enhancement allows an administrator to force a specific bound DN or IP address to unbind. The emergency thread added in the denial of service prevention feature enhances this feature by ensuring that an administrator always has access to unbind bound DNs and IP addresses.

► Group specific search limits

You can now configure "extended" search limits for a defined group of people who are not the administrator or part of the administration group.

► Preservation of operational attributes

The operational attributes `creatorsName`, `createTimestamp`, `lastModifiedBy`, and `lastModifiedTime` are now preserved so that they are consistent between a master and its replicas. In addition, these attributes are now imported by the `ldif2db` and `bulkload` utilities and exported by the `db2ldif` utility.

► Attribute cache

The attribute cache improves search performance for certain search filters by allowing configured attributes and their values to be stored in memory. When a search is performed using a filter that contains all cached attributes and the filter is of a type supported by the attribute cache manager, the filter can be resolved in memory; this leads to improved search performance.

► Serviceability improvements

The following new features improve the serviceability of IBM Tivoli Directory Server:

– Server input and output logging

The actual input and output from the server can now be logged to allow better analysis of problems. In previous releases, the LDAP client library output the BER data to stderr or a file. The new feature adds the capability to record the same formatted BER data one time to the in-memory trace. The trace facility can then be used to extract this data.

– Dynamic trace enablement

Trace information from the server can now be captured without stopping and restarting the server. The level of tracing and the size available for trace output can also be configured dynamically.

► Monitor enhancements

More information has been added to the output of `cn=monitor` to be used in analyzing server performance. These attributes are intended for directory administrators only. The new information includes counts of completed operations by type (for example, `BIND`, `MODIFY`, `COMPARE`, `SEARCH`), depth of the work queue, number of available workers, counts of messages added to the server log, audit log, command-line interface errors, and counts of SSL connections. Information is also included about what worker threads are doing and when they started.

► Additional support on iSeries and pSeries Linux

Support for the new iSeries and pSeries Linux platforms was added in the IBM Tivoli Directory Server 5.1 FixPak 1. IBM Tivoli Directory Server 5.2 adds more support for iSeries and pSeries. The Web Administration Tool can now be used on these platforms, and translated messages have been added.

► System and restricted ACLs - compatibility with OS/390(R)

Support has been added for specification and evaluation of ACLs for the system and restricted attribute classes. This resolves the following interoperability problems between IBM Tivoli Directory Server and OS/390 versions of the LDAP Server.

In previous releases, during replication the IBM Tivoli Directory Server server rejected any directory entry data that contained ACL specifications with references to system or restricted attribute classes. Replication from an OS/390 server provider to an IBM Tivoli Directory Server server consumer therefore failed.

In previous releases, ACL management code could not be written that would run correctly on both types of servers. A client application written for an IBM Tivoli Directory Server environment might not work properly on an OS/390 server because the ACLs might not allow the application to read system attributes. Conversely, a client application developed for an OS/390 server environment would fail to work properly on an IBM Tivoli Directory Server server if the application attempted to set ACLs on system or restricted attributes.

This feature replaces the limited restricted attribute class ACL support, originally provided by IBM Tivoli Directory Server 5.1 Protection of Access Control Information feature (`ibm-slapdACLAccess`), with full directory-specific ACL support. The behavior of this feature is consistent with the existing ACL support provided for the other attribute access classes: Normal, sensitive, and critical.

To maintain consistency with the legacy IBM Tivoli Directory Server ACL model, existing version 5.1 directories that contain entries with explicit ACL specification will be automatically migrated to provide legacy default read, search, and compare access for the subject DN `group:cn=anybody`, as well as any specific access IDs. This is to prevent an unexpected loss of default access after migration. If denial of access is required, it should be explicitly specified in the directory, based on the specific needs and desires of the individual IBM Tivoli Directory Server administrator.

► Support for identity assertions (proxied authentication)

Support has been added for identity assertions, also known as LDAP Proxied Authorization Control. The Proxied Authorization Control allows a client to request that an operation be processed under a provided authorization identity instead of as the current authorization identity associated with the connection.

► Option that the server does not dereference aliases by default

In previous releases, the Java Naming and Directory Interface (JNDI) had dereferencing aliases by default. This sometimes caused performance degradation on the server even if no alias entries existed in the server. A server configuration option has been added to override the dereference option specified in the client search request. Additionally, if no alias objects exist in the directory, the server always bypasses the dereference logic.

► Gateway replication

Gateway replication uses Gateway servers to collect and distribute replication information effectively across a replicating network. The primary benefit of Gateway replication is the reduction of network traffic.

► Enhancements to the Web Administration Tool

Enhancements have been made to the Web Administration Tool, including the following:

– Support for administration of OS/400(R) V5R3 and z/OS(TM) R4 LDAP servers

– Support for object class inheritance from multiple superior objects

– Support for peer-to-peer replication

– Support for gateway replication

– Web Administration support for most new features

4.3 Resources on ITDS

There are several resources available publicly on the Web to find out more information about ITDS. The best place to start is the IBM Tivoli homepage for ITDS at:

http://www-306.ibm.com/software/tivoli/products/directory-server/

From here you can download the product and the most recent Fix Packs and patches, access technical documentation, review recent issues (APARs) that have been resolved, and see published Technotes.

An excellent place for getting questions answered about ITDS is the ITDS NNTP group on IBM's public NNTP service. This group, can be found here:

http://www.news://news.software.ibm.com/ibm.software.ldap

4.4 Summary of ITDS-related chapters

The rest of the chapters in this book go into particular aspects of ITDS installation, configuration, and management. The topics include:

► Chapter 5, "ITDS installation and basic configuration - Windows" on page 95

► Chapter 6, "ITDS installation and basic configuration - AIX" on page 125

► Chapter 7, "ITDS installation and basic configuration on Intel Linux" on page 155

- ► Chapter 8, "IBM Tivoli Directory Server installation - IBM zSeries" on page 185
- ► Chapter 9, "IBM Tivoli Directory Server Distributed Administration" on page 193
- ► Chapter 10, "Client tools" on page 237
- ► Chapter 11, "Schema management" on page 287
- ► Chapter 12, "Group and role management" on page 301
- ► Chapter 13, "Replication" on page 319
- ► Chapter 14, "Access control" on page 395
- ► Chapter 15, "Securing the directory" on page 431
- ► Chapter 16, "Performance Tuning" on page 475
- ► Chapter 17, "Monitoring IBM Tivoli Directory Server" on page 547
- ► Chapter 18, "Debugging IBM Tivoli Directory Server related issues" on page 589
- ► Chapter 19, "Developing C-based applications" on page 603
- ► Appendix A, "DSML Version 2" on page 635
- ► Appendix B, "Directory Integration - IBM Tivoli Directory Integrator" on page 681

5

ITDS installation and basic configuration - Windows

This section describes the installation and basic configuration of ITDS 5.2 on Microsoft Windows NT, Windows 2000, and Windows 2003. For the latest information and updates, as well as code downloads, please check the IBM site at:

http://www-3.ibm.com/software/tivoli/products/directory-server/

ITDS 5.2 has several installation options. You can install using an InstallShield graphical user interface (GUI) or use platform-specific installation methods such as the command line or installation tools for the operating system. This chapter focuses on the GUI installation. For more information on the other types of installation options, please refer to the ITDS product documentation at:

http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html

Before installing, see the *IBM Tivoli Directory Server Version 5.2 Server Readme*, GI11-4151, for any updated information about supported versions of the Microsoft Windows operating system. The readme file is in the root directory of the installation CD or the directory where you unzipped the server package. After installing, the readme file is located in the *installpath*\doc*lang* directory in files server.txt, server.pdf, and server.htm, where:

► *installpath* is the location where the IBM Tivoli Directory Server is installed.

- ► *lang* is the locale you chose when you installed IBM Tivoli Directory Server. For example, for United States English the locale is en_US.

Also see the *IBM Tivoli Directory Server Version 5.2 Readme Addendum,* which contains the latest information. The latest version of the *Readme Addendum* can be found online with the ITDS product documentation:

`http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html`

5.1 Installable components

When you install IBM Tivoli Directory Server, you can install either the client or the server. The server component requires the client.

In addition, you can install the Web Administration Tool on an application server, with or without the server or the client. You can use the Web Administration Tool to administer multiple ITDS servers either locally or remotely. You can install a single Web administration console to manage multiple IBM Tivoli Directory Server servers. You can also manage servers from previous releases, including SecureWay Directory 3.2.x and IBM Directory Server Versions 4.1 and 5.1. See Requirements for the Web Administration Tool in "Web Administration Tool" on page 101 for a complete list of servers that can be managed.

► Client: (Required) Includes a number of key libraries and command utilities required by the server. The client also includes a "C" Development SDK. This component can be installed standalone and requires no other components to be installed. GSKit must be installed if you require SSL for stronger security.

► Server: (Required) The core LDAP server component. You must install at least the client and DB2 in conjunction with the server.

► GSKit: (Optional) Global Security Kit (GSKit) Version 7a is a software package that is required only if Secure Sockets Layer (SSL) Security or Transport Layer Security (TLS) is required.

► IBM WebSphere Express Application Server: (Optional) To use the Web Administration Tool, an application server is required. The embedded version is IBM WebSphere Application Server - Express V5.0.2 is provided with ITDS as an application server.

► Web Administration Tool: (Optional) A Web-based tool used to manage any number of distributed IBM Tivoli Directory Servers as well as prior versions of IBM's Directory Server product line. In order to install the Web Administration tool, you need to have a supported Application Server already installed or the bundled IBM WebSphere Express Application Server is required.

► IBM DB2: (Required) DB2 Universal Database is used as the underling data storage mechanism for the Server.

In order to install the server, at a bare minimum you must install Client, Server, and IBM DB2. If you want to require secure access over SSL to the LDAP Server or Web Administration Tool, you also need to install GSKIT. Finally, if you have not yet installed the Web Administration Tool anywhere else, you will need to install it along with a supported Application Server.

5.2 Installation and configuration checklist

Below you will find an abbreviated checklist that contains a high level summary of the steps required to install and configure ITDS to the point where you can add your own data. Many of these steps are optional but all are recommended to provide a well-tuned, high-performance, and secure directory environment.

ITDS 5.2 installation checklist:

1. Verify that the hardware and operating system meet minimum requirements. See "System and software requirements" on page 99.

2. Obtain products including the latest relevant Fixpacks.

3. Operating system configuration and tuning.

4. Basic product installation. See "Installing the server" on page 102.

5. Add Administrator DN and password. See "Configuring the Administrator DN and password" on page 106.

6. Configure database. See "Configuring the database" on page 108.

7. Add suffix. See "Adding a suffix" on page 115.

8. Tune DB2. See "DB2 tuning" on page 491.

9. Tune slapd parameters in `ibmslapd.conf`. See "Additional slapd and ibmslapd settings" on page 488.

10. Schema customization. See "Modifying the schema" on page 292.

11. Configure ITDS.

    a. TCP/IP Ports ITDS uses.

    b. Password encryption. See "Password encryption" on page 451

    c. Password policy enforcement. See "Password policy enforcement" on page 437.

    d. SSL/TLS, Kerberos, and Digest-MD5. See "SSL/TLS support" on page 455.

    e. Log locations and settings. See "Enabling and disabling the change log" on page 118.

12. Add data.

5.3 System and software requirements

To install the IBM Tivoli Directory Server client and server packages, administer the server, and use the Global Security Kit (GSKit), your computer must meet the minimum system requirements as outlined in this section.

5.3.1 ITDS Client

The IBM Tivoli Directory Server Client SDK provides the tools required to develop LDAP applications as well as a number of the most commonly used command line utilities for manipulating LDAP data within the directory. The following are provided:

► Client libraries that provide a set of C-language APIs

► C header files for building and compiling LDAP applications

► Documentation that describes the programming interface and the sample programs

► Sample programs in source form

► Executable versions of the sample programs:

  – `ldapmodrdn.exe`: LDAP modify relative distinguished name

  – `ldapdelete.exe`: LDAP delete

  – `ldapmodify.exe`: LDAP modify

  – `ldapsearch.exe`: LDAP search

  – `ldapadd.exe`: LDAP add (a renamed version of `ldapmodify`)

  – `ldapchangepwd.exe`: LDAP change password

  – `ldapexop.exe`: LDAP extended operations

The following are the system and software requirements for the ITDS client on Microsoft Windows.

► Operating system requirements

  – Microsoft Windows 2000
  – Microsoft Windows XP
  – Microsoft Windows Server 2003 Standard or Enterprise
  – Microsoft Windows NT 4.0 with Service Pack 6 or later

► Memory requirements

  A minimum of 128 MB RAM is required. For better results, use 256 MB or more.

▶ Disk space requirements

You need at least 100 MB of free space on the disk where you will be installing the client.

5.3.2 ITDS Server (including client)

The Server consists of the following components:

▶ The server executable: `ibmslapd`

▶ Command line import/export utilities

▶ Web-based GUI for administering the directory: Web Administration Tool

▶ Server configuration and database utilities GUI for configuring the directory: Configuration Tool (`ldapxcfg`)

▶ Online Web Administration Tool and Configuration Tool helps

▶ The ITDS Client (see previous section)

The following are the system and software requirements for the ITDS Server on Microsoft Windows. By default, the ITDS Server requires the ITDS client.

▶ Operating system requirements

– Microsoft Windows 2000.

– Microsoft Windows Server 2003 Standard or Enterprise.

– Microsoft Windows NT 4.0 with Service Pack 6 or later. A Microsoft Windows NT file system (NTFS) is required for security support.

▶ Memory requirements

A minimum of 256 MB RAM is required. For better results, use 512 MB or more.

▶ Disk space requirements

– You must have at least 100 MB of free space in the directory specified by the TEMP environment variable.

– You will need 410–610 MB of disk space for the ITDS software on the device you choose to install onto. If IBM DB2 is already installed, then you will need 150 MB to install the other ITDS components.

– Disk space required for data storage is dependent upon the number and size of database entries. Allow a minimum of 80 MB for your database on Windows systems. Also allow another 2 to 3 MB of disk space when creating the DB2 instance.

► Other software requirements

The minimum supported level of IBM DB2 is IBM DB2 Version 7.2 with FixPak 5 or later. DB2 Version 8.1 Enterprise Server Edition with FixPak 2 is included with IBM Tivoli Directory Server and is installed if a supported version of DB2 is not detected on your system. If you have a version of DB2 earlier than Version 7.2 with FixPak 5 installed on your system, you must remove it or upgrade it before installing ITDS. For more information on migrating from previous versions of ITDS, please refer to the Tivoli Software Information Center ITDS 5.2 page at:

http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html

5.3.3 Web Administration Tool

You can install the Web Administration Tool on a computer with or without the client or the server. The Web Administration Tool can be used to administer LDAP servers of the following types:

► IBM Tivoli Directory Server 5.2
► IBM Directory Server 5.1
► IBM Directory Server 4.1
► IBM SecureWay Directory 3.2.2
► OS/400 V5R3
► z/OS R4

Note that for z/OS R4, only the following configurations are supported:

► A single TDBM backend
► A single SDBM backend
► One TDBM and SDBM backend

The Web Administration Tool is supported on the following Microsoft Windows platforms:

► Microsoft Windows NT 4.0
► Microsoft Windows 2000
► Microsoft Windows XP
► Microsoft Windows Server 2003 Standard, Enterprise

To use the Web Administration Tool, you also need the following:

► One of the following application servers:

– The embedded version of IBM WebSphere Application Server - Express V5.0 or later. Version 5.0.2 is provided with IBM Tivoli Directory Server 5.2. (iSeries Linux, pSeries Linux, and HP-UX require version 5.0.2.) If you have version 5.0, which was provided with IBM Directory Server, installed, see the section titled "Migrating the Web Administration Tool and

upgrading the embedded version of WebSphere Application Server - Express" in the *IBM Tivoli Directory Server Installation and Configuration Guide version 5.2*, SC32-1338.

– IBM WebSphere 5.0 or later. (iSeries Linux, pSeries Linux, and HP-UX require version 5.0.2.)

► One of the following Web browsers on the computer from which you will use the Web Administration Tool. (This might or might not be the computer where the Web Administration Tool is installed.)

– On Microsoft Windows platforms

Microsoft Internet Explorer Version 6.0

– On AIX

Mozilla 1.3 or 1.4

– On xSeries® Linux

Mozilla 1.3 or 1.4

– On iSeries, pSeries, zSeries Linux

No browser support available

– On Solaris 7, 8, or 9

Mozilla 1.3 or 1.4

– On HP-UX

Mozilla 1.3 or 1.4

5.4 Installing the server

Use the information in the following sections to install ITDS 5.2 on a Windows platform using the Installshield GUI.

> **Note:** The following installation instructions do not cover migration scenarios. For information on how to migrate previous versions of the Directory Server to ITDS 5.2, please refer to the *IBM Tivoli Directory Server Installation and Configuration Guide version 5.2*, SC32-1338.

5.4.1 Create a user ID for ITDS

Before you install, create or be sure that you have created the user ID that will own ITDS's IBM DB2 database used to store the directory data. You will be asked to provide this user ID and its password during configuration, which runs automatically after installation and system restart. The user ID must be 8

characters or less, and it must be a member of the Administrators group. If you are creating a new database, a DB2 instance with the same name as the user ID will be created to hold the database.

The method used to create the user varies from one Microsoft Windows operating system to another. Please refer to the operating system documentation for more details on this process.

> **Tip:** A simple way to create the type of user account that ITDS requires on a Microsoft Windows 2000 Server is with the following two commands. This example uses a username of `ldapdb2` and a password of `somepassword`. Enter these two commands at a Microsoft Windows command prompt window (as an Administrator).
>
> ```
> NET USER ldapdb2 somepassword /ADD /ACTIVE:yes /expires:never /comment:"ITDS
> Account"
>
> NET LOCALGROUP Administrators /add ldapdb2
> ```
>
> The account `ldapdb2` now exists on the Windows Server, is active, and has the proper privileges. You can now move into the actual setup of ITDS.

5.4.2 Installing ITDS with the Installshield GUI

To install:

1. On the computer where you are installing the IBM Tivoli Directory Server, stop any programs that are running and close all windows. If you have open windows, the initial IBM Tivoli Directory Server installation window might be hidden behind other windows.

2. If you are installing from a CD, insert the CD in your CD-ROM drive.

3. If you are installing locally from a CD or remotely from the network, go to the drive for your CD-ROM or for the appropriate network path. If you downloaded a zipped file, go to the directory where you unzipped the file.

4. In the \ismp folder, double-click the **setup.exe** icon. The language window is displayed.

> **Note:** When installing on Windows, if the installation program exits without displaying the language window, it might be caused by one of the following:
>
> ► Backlevel video drivers. Update your video drivers to the most recent levels to correct this.
>
> ► Not enough space in the directory specified by the TEMP environment variable. Be sure that you have at least 100 MB of free space in this directory.

5. Select the language you want to use during IBM Tivoli Directory Server installation. Click **OK**.

> **Note:** This is the language used in the installation program, not in IBM Tivoli Directory Server. You choose the language used in IBM Tivoli Directory Server in step 10.

6. On the Welcome window, click **Next**.

7. After reading the Software license agreement, select **I accept the terms in the license agreement**. Click **Next**.

8. Any preinstalled components and corresponding version levels are displayed. Click **Next**.

9. To install to the default directory, click **Next**. You can specify a different directory by clicking **Browse**.

> **Note:** Do not use special characters, such as hyphen (-) and period (.) in the name of the installation directory. If you do not use the default location, use a name such as ldap or ldapdir. Do not use a name such as ldap-dir or ldap.dir.

10. Select the language you want to use in IBM Tivoli Directory Server 5.2. Click **Next**.

11. A window showing the following components for installation is displayed, as shown in Figure 5-1 on page 105:

   – Client SDK 5.2
   – Web Administration Tool 5.2
   – Server 5.2
   – IBM WebSphere Application Server - Express 5.0.2
   – DB2 V8.1
   – GSKit

The components that are not yet installed are preselected. You can choose to reinstall the server, the client, or the Web Administration Tool if they were previously installed.
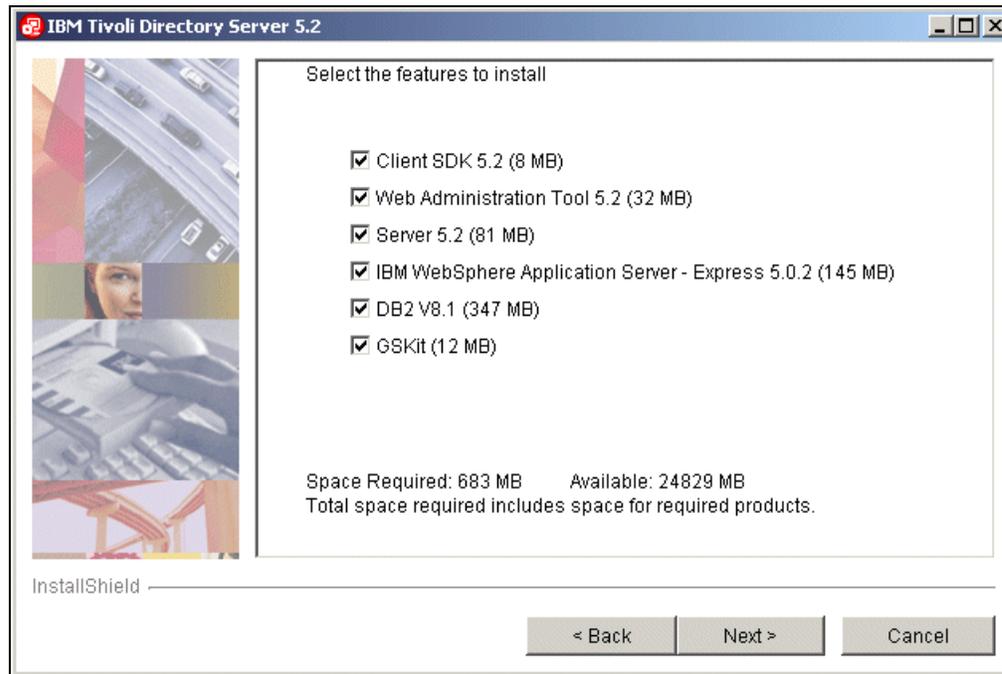
Figure 5-1   Install component selection window

Figure 5-1 also indicates the amount of disk space required and available on the selected drive.

Be sure the components you want to install are selected, and click **Next**.

12. If you selected DB2 V8.1 in step 12, a window is displayed prompting you to enter a Windows user ID and password for the DB2 system ID. The default user ID is db2admin. On the window:

   a. Type the user ID or accept the default.

   b. Type the password, and then type the password again for verification.

   c. Click **Next**.

> **Note:** Note the following:
>
> - ▶ This user ID must not be the one you created in `Creating the DB2 database owner`.
>
> - ▶ If you are using an existing Microsoft Windows user ID, be sure that your password is correct. Otherwise, DB2 does not install correctly.
>
> - ▶ If you are using an existing Windows user ID, it must be a member of the Administrators group.
>
> - ▶ If you are not using an existing user ID, DB2 creates the user ID you specify with the password you type.

13. The installation program now has enough information to begin installing. A summary window displays the components you selected and the locations where the selected components will be installed. Click **Back** to change any of your selections. Click **Next** to begin installation.

14. After the files are installed:

    – If you installed the client, the Client Readme file is displayed. Read the file and click **Next**.

    – If you installed the server, the server Readme file is also displayed. Read the file and click **Next**.

    – If you installed the Web Administration Tool, the Web Administration Tool Readme file is also displayed. Read the file and click **Next**.

15. Select to restart your computer now or later. Click **Finish**.

> **Note:** If you installed the server, you must restart your system to complete IBM Tivoli Directory Server configuration. You are unable to use IBM Tivoli Directory Server until this is completed.

After your computer is restarted, if you installed the server, log in using the same user ID that you used to install IBM Tivoli Directory Server. The Configuration Tool automatically runs so that you can complete the server configuration. Before you can use the server, you must set the administrator DN and password and configure the database that will store the directory data.

5.4.3 Configuring the Administrator DN and password

Each ITDS Server has a special "super-user" account associated with it that provides maximum privileges within ITDS. You will need to create this account before you can administer ITDS.

To set the administrator DN and password, refer to Figure 5-2 on page 108, and perform these steps:

1. In the IBM Tivoli Directory Server Configuration Tool window, click **Administrator DN/password** in the task list on the left.

2. In the Administrator DN/password window on the right, type a valid DN (or accept the default DN, cn=root) in the Administrator DN field.

   The IBM Directory Server administrator DN is the DN used by the administrator of the directory. This administrator is the one user who has full access to all data in the directory.

   The default DN is cn=root. DNs are not case sensitive. If you are unfamiliar with X.500 format, or if for any other reason you do not want to define a new DN, accept the default DN.

3. Type the password for the Administrator DN in the Administrator Password field. You must define a password. Passwords are case-sensitive.

   Record the password for future reference.

   Note: Double byte character set (DBCS) characters in the password are not supported.

4. Retype the password in the Confirm password field.

5. Click **OK**.

Figure 5-2   Setting the administrator DN and password

5.4.4 Configuring the database

Since ITDS uses IBM DB2 as the storage repository for all data, prior to adding data to your directory, you will need to configure a database instance that will be associated with ITDS.

To configure the directory database:

1. Before you configure the database that ITDS will use, create or be sure that you have previously created a valid user ID that will own the DB2 database used to store the directory data. You will be asked to provide this user ID and its password during configuration, which runs automatically after installation and system restart. The user ID must be 8 characters or less, and it must be a member of the Administrators group. If you are creating a new database, a DB2 instance with the same name as the user ID will be created to hold the database.

> **Note:** Verify that the user ID you have created or assigned can successfully log into the system. Check to ensure the password does not expire on first login. Check to see if the account is enabled.

2. In the Configuration Tool, click **Configure database** in the task list on the left. Select **Configure new database** and click **Next** as shown in Figure 5-3 on page 109.
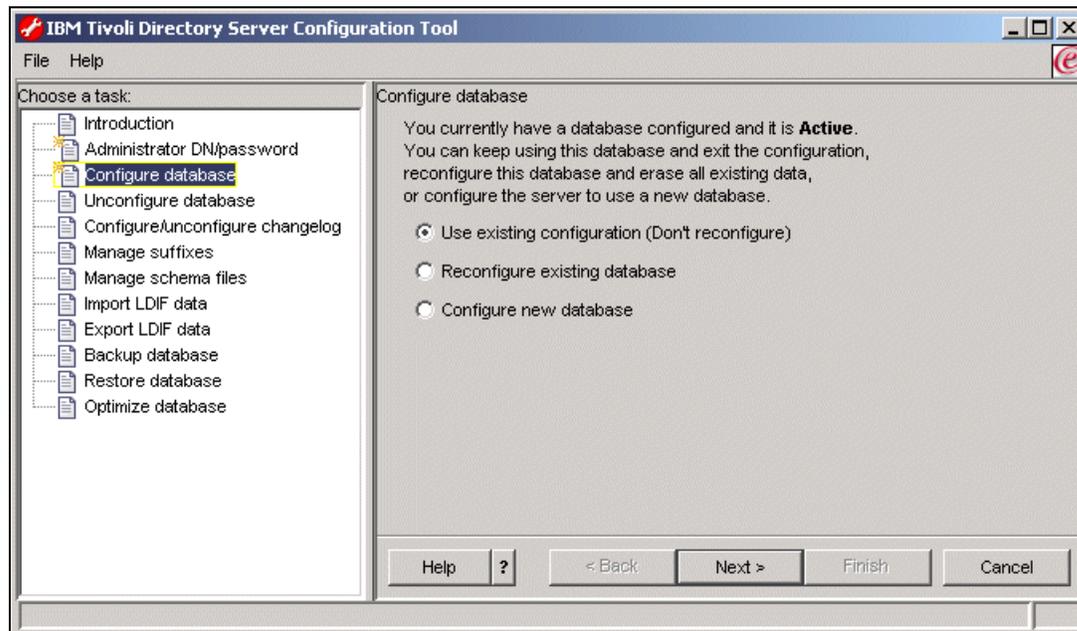
Figure 5-3   Database configuration

3. A user ID and password is requested, as shown in Figure 5-4 on page 110:

    a. Type a user ID in the User ID field. This user ID must already exist before you can configure the database. This is the user ID you created in step 1. Type a password for the user in the Password field. Passwords are case-sensitive.

    b. Click **Next**.
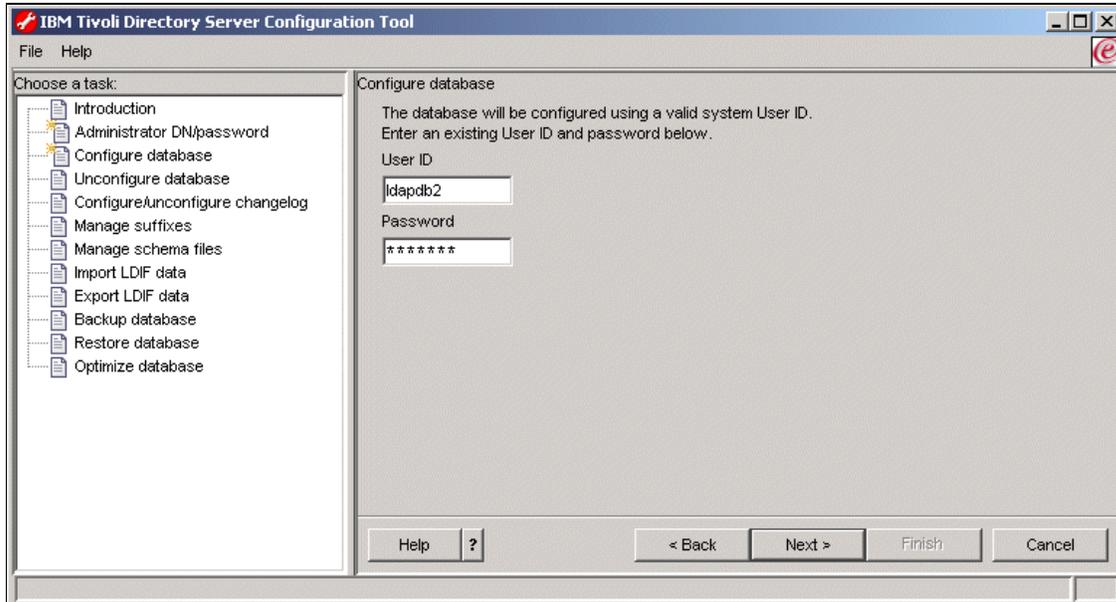
Figure 5-4   Database configuration - Setting the user ID and password for the database

4. Next you will be prompted for a name for the database, as shown in Figure 5-5 on page 111:

   a. Type the name you want to give the DB2 database. The name can be from 1 to 8 characters long. The database will be created in an instance with the same name as the user ID.

   b. Click **Next**.
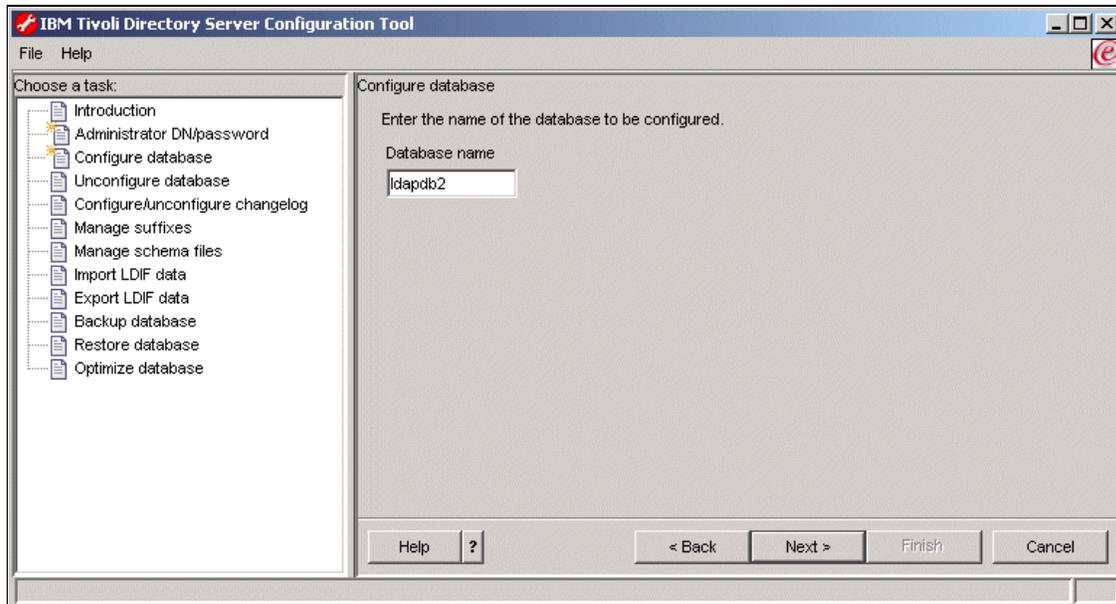
Figure 5-5   Database configuration - Choose DB2 database name

5. If the database location is requested, as shown in Figure 5-6 on page 112:

   a. Type the location for the database in the Database location field. For Windows platforms, this must be a drive letter.

   Be sure that you have at least 80 MB of free hard disk space in the location you specify and that additional disk space is available to accommodate growth as new entries are added to the directory.

   b. Click **Next**.
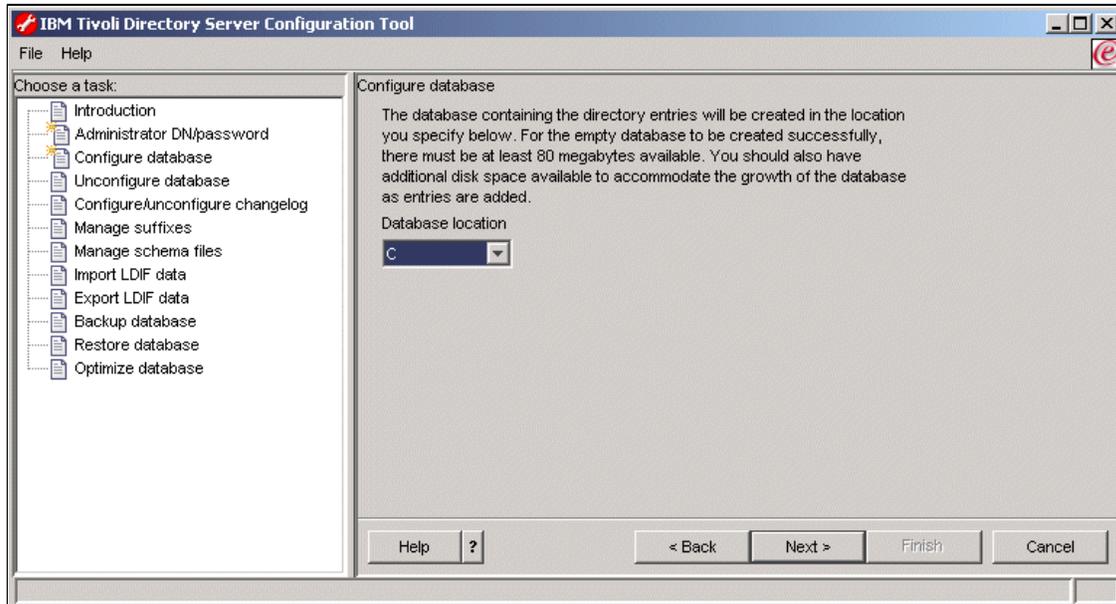
Figure 5-6   Database configuration - Choosing an install location (Windows)

6. If a character set selection is requested, as shown in Figure 5-7 on page 113:

   a. Click the type of database you want to create. You can create a UCS
      Transformation Format (UTF-8) database, in which LDAP clients can store
      UTF-8 character data, or a local code page database, which is a database
      in the local code page.

> **Note:** IBM Tivoli Directory Server supports a wide variety of national language characters through the UTF-8 (UCS Transformation Format) character set. As specified for the LDAP Version 3 protocol, all character data that is passed between an LDAP client and a server is in UTF-8. Consequently, the directory server can be configured to store any national language characters that can be represented in UTF-8. The limitations on what types of characters can be stored and searched for are determined by how the database is created. The database character set can be specified as UTF-8 or it can be set to use the server system's local character set (based on the locale, language, and code page environment).
>
> If you specify UTF-8, you can store any UTF-8 character data in the directory. LDAP clients running anywhere in the world (in any UTF-8 supported language) can access and search the directory. In many cases, however, the client has limited ability to properly display the results retrieved from the directory in a particular language/character set. There is also a performance advantage to using a UTF-8 database because no data conversion is required when storing data to or retrieving data from the database.

    b. Click **Next**.

Figure 5-7   Database configuration - Codepage selection

7. In the verification window, shown in Figure 5-8, information is displayed about the configuration options you specified. To return to an earlier window and change information, click **Back**. To begin configuration, click **Finish**.
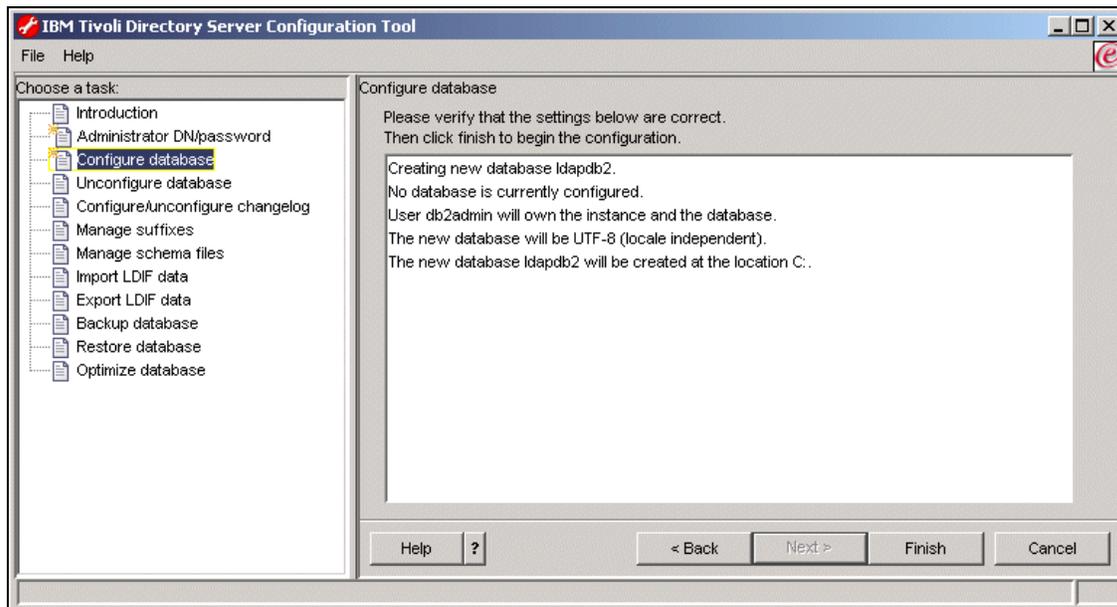
Figure 5-8   Configuration final confirmation

8. The completion window is displayed, as shown in Figure 5-9 on page 115. Click **Close**.
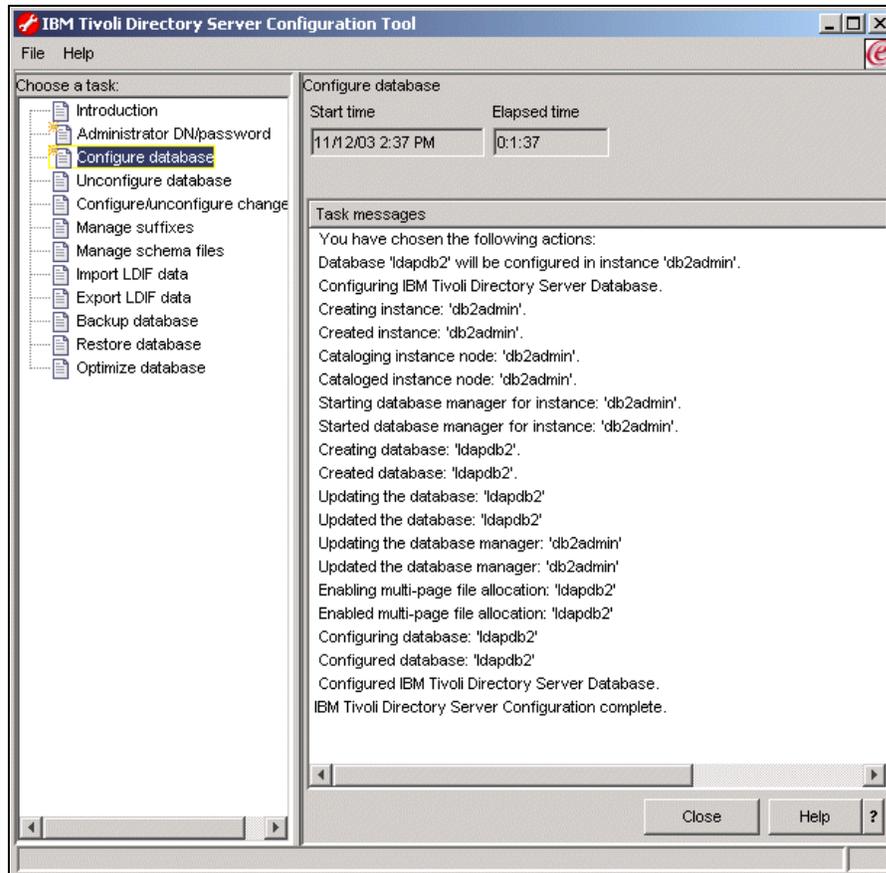
Figure 5-9   Database configuration - Results window

5.4.5 Adding a suffix

A suffix (also known as a naming context) is a distinguished name (DN) that identifies the top entry in a locally held directory hierarchy. Because of the relative naming scheme used in LDAP, this DN is also the suffix of every other entry within that directory hierarchy. A directory server can have multiple suffixes, each identifying a locally held directory hierarchy, for example, o=ibm,c=us.

Entries to be added to the directory must have a suffix that matches the DN value, such as ou=Marketing,o=ibm,c=us. If a query contains a suffix that does not match any suffix configured for the local database, the query is referred to the LDAP server that is identified by the default referral. If no LDAP default referral is

specified, an `Object does not exist` result is returned. The server must be stopped before you add or remove suffixes.

Add a suffix

To add a suffix:

1. In the Configuration Tool, click **Manage suffixes** in the task list on the left, as shown in Figure 5-10.

2. In the Manage suffixes window, type the suffix you want to add in the SuffixDN field, and click **Add**.

3. When you have added all the suffixes you want, click **OK**. When you click **Add**, the suffix is added to the list in the current suffix DNs box; however, the suffix is not actually added to the directory until you click **OK**.
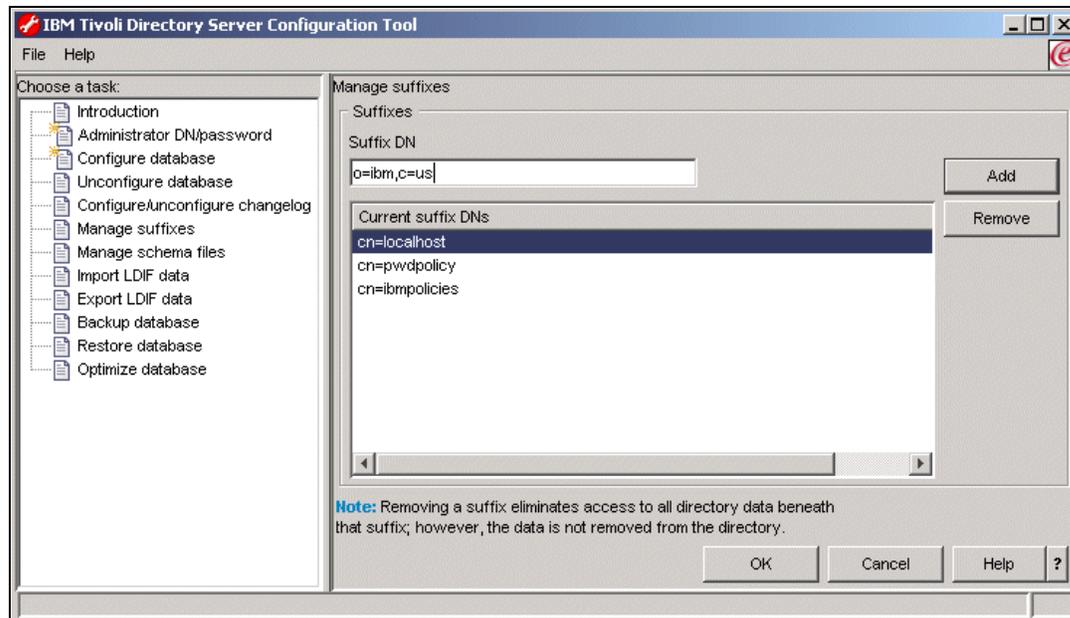
Figure 5-10   Adding a suffix

Removing a suffix

To remove a suffix:

1. In the Configuration Tool, click **Manage suffixes** in the task list on the left.

2. In the Manage suffixes window, click the suffix you want to remove in the Current suffix DNs box, and click **Remove**.

3. When you have selected all the suffixes you want to remove, click **OK**. When you click **Remove**, the suffix is removed from the list in the current suffix DNs box; however, the suffix is not actually removed until you click **OK**.

5.4.6 Removing or reconfiguring a database

At some point you may need to remove the IBM DB2 database instance that is associated with ITDS. The ITDS ldapxcfg tool allows you to *unconfigure* the database instance, *unconifgure* and *destroy* the database instance, and *unconfigure*, *destroy*, and *delete* the database instance.

To unconfigure the database:

1. In the Configuration Tool, click **Unconfigure database** in the task list on the left.

2. In the Unconfigure database window, click of the following:

   – Unconfigure only

   Does not destroy any existing LDAP DB2 data. However, the configuration information for the database will be removed from the configuration file (ibmslapd.conf), and the database will be inaccessible to the directory server.

   – Unconfigure and destroy database

   Removes the existing database and its contents, and removes the configuration information for the database from the configuration file.

   – Unconfigure and destroy database and delete instance

   Removes the existing database and its contents, removes the configuration information for the database from the configuration file, and deletes the instance in which the database is located.

3. Click **Unconfigure**.

Figure 5-11   Unconfiguring the DB2 database associated with ITDS

Once you have completed these steps, you may now *configure* or *re-configure* a
new database instance for use with ITDS. See "Configuring the database" on
page 108 for more information.

5.4.7 Enabling and disabling the change log

The change log database is used to record changes to the schema or directory
entries in the typical LDAP entry structure that can be retrieved through the
LDAP API. The change log records all update operations: *Add*, *delete*, *modify,*
and *modrdn*. The change log enables LDAP client applications to retrieve a set of
changes that have been made to an IBM Tivoli Directory Server database. The
client might then update its own replicated or cached copy of the data.

The change log function causes all updates to LDAP to be recorded in a
separate change log DB2 database (that is, a different database from the one
used to hold the LDAP server Directory Information Tree). The change log
database can be used by other applications to query and track LDAP updates.
The change log function is disabled by default.

Unlike some other directory servers on the market, the change log is not required by ITDS to setup replication. Typically, the change log is enabled so meta-directory sychronization products such as IBM Tivoli Directory Integrator (ITDI) can detect changes occurring within ITDS and then push those changes to other non-ITDS data repositories.

There are some performance considerations when you enable the change log since all changes within ITDS are now logged to a separate a database instance. You should evaluate the impact of enabling the change log during in the pre-deployment phases of your ITDS deployment.

You can use the `ldapxcfg` Configuration Tool to enable or disable the change log. The server must be stopped before you enable or disable the change log.

To enable the change log, refer to Figure 5-12 on page 120 and perform the following steps:

1. In the Configuration Tool, click **Configure/unconfigure changelog** in the task list on the left.

2. In the Configure/unconfigure changelog window, select the **Enable change log database** check box.

3. In the Maximum number of log entries box, click **Unlimited** if you want an unlimited number of entries in the change log. If you want to limit the number of entries, click Entries and type the maximum number of entries you want recorded. The default is 1,000,000 entries.

4. In the Maximum age box, accept the default of Unlimited if you want entries to remain in the change log indefinitely, or click **Age** and type the number of days and hours for which you want each entry to be kept.

5. Click **Update**.

Figure 5-12   Enabling the change log

To disable the change log:

1. In the Configuration Tool, click **Configure/unconfigure changelog** in the task list on the left.

2. In the Configure/unconfigure changelog window, clear the Enable change log database check box.

3. Click **Update**.

5.5 Starting ITDS

There are a number of other optional tasks you can perform within the Directory Configuration tool at this point such as adding custom schema and importing data. Those tasks do not have to be completed before you initially start the server. Those topics are covered in subsequent chapters.

The easiest way to start the server is by typing `ibmslapd` in a windows command prompt, as shown in Example 5-1.

Example 5-1   Starting the Directory Server

```
C:\>ibmslapd

Dec 13 16:01:43 2003  Server starting.
Dec 13 16:01:44 2003  Plugin of type EXTENDEDOP is successfully loaded from libevent.dll.
Dec 13 16:01:44 2003  Plugin of type EXTENDEDOP is successfully loaded from libtranext.dll.
Dec 13 16:01:45 2003  Plugin of type EXTENDEDOP is successfully loaded from libldaprepl.dll.
Dec 13 16:01:45 2003  Plugin of type PREOPERATION is successfully loaded from libDSP.dll.
Dec 13 16:01:45 2003  Plugin of type PREOPERATION is successfully loaded from libDigest.dll.
Dec 13 16:01:45 2003  Plugin of type EXTENDEDOP is successfully loaded from libevent.dll.
Dec 13 16:01:45 2003  Plugin of type EXTENDEDOP is successfully loaded from libtranext.dll.
Dec 13 16:01:45 2003  Plugin of type AUDIT is successfully loaded from C:/Program
Files/IBM/LDAP/bin/libldapaudit.dll.
Dec 13 16:01:45 2003  Plugin of type PREOPERATION is successfully loaded from C:/Program
Files/IBM/LDAP/bin/libcl.dll.
Dec 13 16:01:45 2003  Plugin of type EXTENDEDOP is successfully loaded from libevent.dll.
Dec 13 16:01:45 2003  Plugin of type EXTENDEDOP is successfully loaded from libtranext.dll.
Dec 13 16:01:45 2003  Plugin of type DATABASE is successfully loaded from C:/Program
Files/IBM/LDAP/bin/libback-rdbm.dll.
Dec 13 16:01:45 2003  Plugin of type REPLICATION is successfully loaded from C:/Program
Files/IBM/LDAP/bin/libldaprepl.dll.
Dec 13 16:01:45 2003  Plugin of type EXTENDEDOP is successfully loaded from C:/Program
Files/IBM/LDAP/bin/libback-rdbm.dll.
Dec 13 16:01:45 2003  Plugin of type PREOPERATION is successfully loaded from C:/Program
Files/IBM/LDAP/bin/libcl.dll.
Dec 13 16:01:45 2003  Plugin of type EXTENDEDOP is successfully loaded from libevent.dll.
Dec 13 16:01:45 2003  Plugin of type DATABASE is successfully loaded from C:/Program
Files/IBM/LDAP/bin/libback-rdbm.dll.
Dec 13 16:01:45 2003  Plugin of type PREOPERATION is successfully loaded from C:/Program
Files/IBM/LDAP/bin/libcl.dll.
Dec 13 16:01:45 2003  Plugin of type EXTENDEDOP is successfully loaded from libevent.dll.
Dec 13 16:01:45 2003  Plugin of type DATABASE is successfully loaded from C:/Program
Files/IBM/LDAP/bin/libback-config.dll.
Dec 13 16:01:50 2003  Plugin of type EXTENDEDOP is successfully loaded from libloga.dll.
Dec 13 16:01:50 2003  Non-SSL port initialized to 389.
Dec 13 16:01:54 2003  IBM Tivoli Directory (SSL), Version 5.2Server started.
Dec 13 16:01:54 2003  Started 15 worker threads to handle client requests.

C:\>
```

After you type `ibmslapd` at the command prompt, a number of messages will be logged to the screen. One of them should say `IBM Tivoli Directory (SSL) Version 5.2 Server started`.

To verify ITDS is indeed running, configured properly, and responding to queries, you can type the following command at the Windows command prompt:

```
ldapsearch -s base -b "" objectclass=*
```

The output of this command is shown in Example 5-2.

Example 5-2   Querying the root DSE

```
C:\>ldapsearch -s base -b "" objectclass=*

namingcontexts=CN=SCHEMA
namingcontexts=CN=LOCALHOST
namingcontexts=CN=PWDPOLICY
namingcontexts=CN=IBMPOLICIES
namingcontexts=O=IBM,C=US
namingcontexts=CN=CHANGELOG
subschemasubentry=cn=schema
supportedextension=1.3.18.0.2.12.1
supportedextension=1.3.18.0.2.12.3
supportedextension=1.3.18.0.2.12.5
supportedextension=1.3.18.0.2.12.6
supportedextension=1.3.18.0.2.12.15
supportedextension=1.3.18.0.2.12.16
supportedextension=1.3.18.0.2.12.17
supportedextension=1.3.18.0.2.12.19
supportedextension=1.3.18.0.2.12.44
supportedextension=1.3.18.0.2.12.24
supportedextension=1.3.18.0.2.12.22
supportedextension=1.3.18.0.2.12.20
supportedextension=1.3.18.0.2.12.28
supportedextension=1.3.18.0.2.12.30
supportedextension=1.3.18.0.2.12.26
supportedextension=1.3.6.1.4.1.1466.20037
supportedextension=1.3.18.0.2.12.35
supportedextension=1.3.18.0.2.12.40
supportedextension=1.3.18.0.2.12.46
supportedextension=1.3.18.0.2.12.37
supportedcontrol=2.16.840.1.113730.3.4.2
supportedcontrol=1.3.18.0.2.10.5
supportedcontrol=1.2.840.113556.1.4.473
supportedcontrol=1.2.840.113556.1.4.319
supportedcontrol=1.3.6.1.4.1.42.2.27.8.5.1
supportedcontrol=1.2.840.113556.1.4.805
```

```
supportedcontrol=2.16.840.1.113730.3.4.18
supportedcontrol=1.3.18.0.2.10.15
supportedcontrol=1.3.18.0.2.10.18
security=none
port=389
supportedsaslmechanisms=CRAM-MD5
supportedsaslmechanisms=DIGEST-MD5
supportedldapversion=2
supportedldapversion=3
ibmdirectoryversion=5.2
changelog=cn=changelog
firstchangenumber=1
lastchangenumber=1
ibm-ldapservicename=TEST-WIN2K
ibm-serverId=718b8a13-a75f-4e2e-acb7-e8aa69095157
ibm-supportedacimechanisms=1.3.18.0.2.26.3
ibm-supportedacimechanisms=1.3.18.0.2.26.4
ibm-supportedacimechanisms=1.3.18.0.2.26.2
vendorname=International Business Machines (IBM)
vendorversion=5.2
ibm-sslciphers=N/A
ibm-slapdisconfigurationmode=FALSE
ibm-slapdSizeLimit=500
ibm-slapdTimeLimit=900
ibm-slapdDerefAliases=always
ibm-supportedAuditVersion=2
ibm-sasldigestrealmname=TEST-WIN2K

C:\>
```

If the suffix you added in "Adding a suffix" on page 115 is displayed in the output of your ldapsearch command in the format `namingcontexts=O=IBM,C=US` (o=ibm,c=us is the suffix added in this example), then ITDS's slapd LDAP listener is configured properly and open for business.

6

ITDS installation and basic configuration - AIX

This section describes the installation and basic configuration of ITDS 5.2 on the IBM AIX operating system. For the latest information and updates, as well as code downloads, please check the IBM site at:

http://www-3.ibm.com/software/tivoli/products/directory-server/

ITDS 5.2 has several installation options. You can install using an InstallShield graphical user interface (GUI) or use platform-specific installation methods such as the command line or installation tools for the operating system. This chapter focuses on the GUI installation. For more information on the other types of installation options, please refer to the ITDS documentation at:

http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html

Before installing, see *IBM Tivoli Directory Server Version 5.2 Server Readme*, GI11-4151, for any updated information about supported versions of the AIX operating system. The readme file is in the root directory of the CD or the directory where you extracted the server package from the tape archive (tar) image. After installing, the readme file is located in the *installpath*\doc*lang* directory in files server.txt, server.pdf, and server.htm, where:

► *installpath* is the location where the IBM Tivoli Directory Server is installed.

► *lang* is the locale you chose when you installed IBM Tivoli Directory Server; for example, for United States English the locale is en_US.

Also see the *IBM Tivoli Directory Server Version 5.2 Readme Addendum,* which contains the latest information. The latest version of the *Readme Addendum* can be found online with the ITDS product documentation:

`http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html`

6.1 Installable components

When you install IBM Tivoli Directory Server, you can install either the client or the server, which requires the client.

In addition, you can install the Web Administration Tool on an application server, with or without the server or the client. You can use the Web Administration Tool to administer IBM Tivoli Directory Server servers either locally or remotely. You can install a single Web Administration console to manage multiple IBM Tivoli Directory Server servers. You can manage servers from previous releases, including SecureWay Directory 3.2.x and IBM Directory Server Versions 4.1 and 5.1. See Requirements for the Web Administration Tool in "Web Administration Tool" on page 132 for a complete list of servers that can be managed.

► Client: (Required) Includes a number of key libraries and command utilities required by the server. The client also includes a "C" Development SDK. This component can be installed standalone and requires no other components to be installed. GSKit must be installed if you require SSL for stronger security.

► Server: (Required) The core LDAP server component. You must install at least the client and DB2 in conjunction with the server.

► IBM GSKit: (Optional) IBM Global Security Kit (GSKit) Version 7a is a software package that is required only if Secure Sockets Layer (SSL) Security or Transport Layer Security (TLS) is required.

► IBM WebSphere Express Application Server: (Optional) To use the Web Administration Tool, an application server is required. The embedded version of IBM WebSphere Application Server - Express V5.0.2 is provided with ITDS as an application server.

► Web Administration Tool: (Optional) A Web-based tool used to manage any number of distributed IBM Tivoli Directory Servers as well as prior versions of IBM's Directory Server product line. In order to install the Web Administration tool, you will need to have a supported application server already installed or the bundled IBM WebSphere Express Application Server is required.

► DB2: (Required) IBM DB2 Universal Database is used as the underling data storage mechanism for the server.

In order to install the server, at a bare minimum you must install client, server, and DB2. If you want to require secure access over SSL to the LDAP Server or Web Administration Tool, you will also need to install GSKIT. Finally, if you have not yet installed the Web Administration Tool anywhere else, you will need to install it along with a supported application server.

6.2 Installation and configuration checklist

Below you will find an abbreviated checklist that contains a high-level summary of the steps required to install and configure ITDS to the point where you can add your own data. Many of these steps are optional but all are recommended in order to provide a well-tuned, high-performance, and secure directory service environment.

ITDS 5.2 installation checklist:

1. Verify that the hardware and operating system meet minimum requirements. See "System and software requirements" on page 129.

2. Obtain product including latest relevant Fixpacks.

3. Operating system configuration and tuning.

4. Basic product installation. See "Installing the server" on page 133.

5. Add Administrator DN and password. See "Configuring the Administrator DN and password" on page 137.

6. Configure database. See "Configuring the database" on page 138.

7. Add suffix. See "Adding a suffix" on page 145.

8. Tune DB2. See "DB2 tuning" on page 491.

9. Tune `slapd` parameters in `ibmslapd.conf`. See "Additional slapd and ibmslapd settings" on page 488.

10. Schema customization. See "Modifying the schema" on page 292.

11. Configure ITDS.

    c. TCP/IP Ports ITDS uses.

    d. Password encryption. See "Password encryption" on page 451.

    e. Password policy enforcement. See "Password policy enforcement" on page 437.

    f. SSL / TLS, Kerberos, and Digest-MD5. See "SSL/TLS support" on page 455.

    g. Log locations and settings. See "Enabling and disabling the change log" on page 148.

12. Add data.

6.3 System and software requirements

To install the IBM Tivoli Directory Server client and server packages, administer the server, and use the Global Security Kit (GSKit), your computer must meet the minimum system requirements as outlined in this section.

6.3.1 ITDS Client

The IBM Tivoli Directory Server Client SDK provides the tools required to develop LDAP applications as well as a number of the most commonly used command line utilities for manipulating LDAP data within the directory. The following are provided:

► Client libraries that provide a set of C-language APIs

► C header files for building and compiling LDAP applications

► Documentation that describes the programming interface and the sample programs

► Sample programs in source form

► Executable versions of the sample programs

– `ldapmodrdn`: LDAP modify relative distinguished name
– `ldapdelete`: LDAP delete
– `ldapmodify`: LDAP modify
– `ldapsearch`: LDAP search
– `ldapadd`: LDAP add (a renamed version of `ldapmodify`)
– `ldapchangepwd`: LDAP change password
– `ldapexop.exe`: LDAP extended operations

The following are the system and software requirements for the ITDS client on AIX. The client is 32-bit and does not require 64-bit support if installed on a different machine than the ITDS Server component.

► Operating system requirements

– IBM AIX 4.3.3. (The GUI Install is not supported on AIX 4.3.3. Please refer to the *IBM Tivoli Directory Server Version 5.2 Installation & Configuration Guide*, SC32-1338, for alternative installation methods.)

– IBM AIX 5.1.

– IBM AIX 5.2.

► Memory requirements

A minimum of 128 MB RAM is required. For better results, use 256 MB or more.

- Disk space requirements

  You must have at least 100 MB of free space in the /var directory and at least 200 MB of free space in the /tmp directory.

- Other requirements

  The following additional requirements may apply:

  – The Korn shell is required.

  – For AIX 4.3.3 you must install IBM AIX Maintenance Level 8 or later. On AIX 5.1, you must install IBM AIX Maintenance Level 4 or later. On AIX 5.2, you must install IBM AIX Maintenance Level 1 or later.

  – The bos.loc.iso.ZH_TW fileset must be installed for the Taiwan locale. The fileset is available from the IBM AIX 4.3.3 installation medium.

  – The xlC.rte 6.0.0.0 or later fileset is required for GSKit 7a on AIX 5.1 and 5.2.

  – The xlC.aix43.rte 6.0.0.0 or later fileset is required for GSKit 7a on AIX 4.3.3.

  – To use GSKit, the IBM JRE or JDK 1.4.1 or an equivalent JRE or JDK is required.

6.3.2 ITDS Server (including client)

The server consists of the following components:

- The server executable: `ibmslapd`

- Command line import and export utilities

- Web-based GUI for administering the directory: Web Administration Tool

- Server configuration and database utilities GUI for configuring the directory: Configuration Tool (`ldapxcfg`)

- Online Web Administration Tool and Configuration Tool helps

- The ITDS Client

The following are the system and software requirements for the ITDS Server on AIX. By default, the ITDS Server requires the ITDS client. You must be running on 64-bit hardware and have 64-bit AIX kernel installed.

> **Tip:** To verify that your AIX hardware is 64-bit, run the following command:
>
> ```
> bootinfo -y
> ```
>
> If the command returns 32, your hardware is 32-bit.
>
> In addition, if you type the command `lsattr -El proc0`, the output of the command returns the type of processor for your server. If you have any of the following, you have 64-bit hardware: RS64 I, II, III, IV, POWER3™, POWER3 II or POWER4™.
>
> To verify that you have the 64 bit kernel (/usr/lib/boot/unix_64) installed and running, run the following command:
>
> ```
> bootinfo -K
> ```
>
> Go to http://www-1.ibm.com/support/docview.wss?uid=isg1hintsTips0214 for more information on determining if you system has 64-bit hardware and/or a 64-bit kernel.

The requirements are:

► Operating system requirements

– IBM AIX 5.1
– IBM AIX 5.2

► Memory requirements

A minimum of 512 MB RAM is required. For better results, have 1 GB or more available.

► Disk space requirements

– You must have at least 100 MB of free space in the /var directory and at least 400 MB in the /tmp directory.

– You will need 460–660 MB of disk space for the ITDS software on the device you choose to install on. If DB2 is already installed, then you will need 160 MB to install the other ITDS components.

– Disk space required for data storage is dependent upon the number and size of database entries. Allow a minimum of 80 MB for your database on AIX systems. Also, ensure that there is approximately another 4 MB of disk space in the home directory of the user who will own the database to create the DB2 instance.

► Other software

– The Korn shell is required.

– On AIX 5.1, you must install IBM AIX Maintenance Level 4 or later. On AIX 5.2, you must install IBM AIX Maintenance Level 1 or later.

– The xlC.aix50.rte 6.0.0.0 or later fileset is required for GSKit 7a.

– To use GSKit, the IBM JRE or JDK 1.4.1 or an equivalent JRE or JDK is required.

– IBM DB2 Universal Database for AIX Version 8.1 Enterprise Server Edition with FixPak 2 (DB2) is included with the IBM Tivoli Directory Server. For AIX, no previous versions of DB2 are supported.

6.3.3 Web Administration Tool

You can install the Web Administration Tool on a computer with or without the client or the server. The Web Administration Tool can be used to administer LDAP servers of the following types:

► IBM Tivoli Directory Server 5.2
► IBM Directory Server 5.1
► IBM Directory Server 4.1
► IBM SecureWay Directory 3.2.2
► IBM OS/400 V5R3
► IBM z/OS R4

Note that for z/OS R4, only the following setups are supported:

► A single TDBM backend
► A single SDBM backend
► One TDBM and SDBM backend

The Web Administration Tool is supported on the following versions of AIX:

► IBM AIX 4.3.3
► IBM AIX 5.1
► IBM AIX 5.2

To use the Web Administration Tool, you also need the following:

► One of the following application servers:

– The embedded version of IBM WebSphere Application Server - Express V5.0 or later. Version 5.0.2 is provided with IBM Tivoli Directory Server 5.2. (iSeries Linux, pSeries Linux, and HP-UX require version 5.0.2.) If you have version 5.0, which was provided with IBM Tivoli Directory Server, installed, see the section titled "Migrating the Web Administration Tool and upgrading the embedded version of WebSphere Application Server - Express" in the *IBM Tivoli Directory Server Installation and Configuration Guide version 5.2*, SC32-1338.

- IBM WebSphere 5.0 or later. (iSeries Linux, pSeries Linux, and HP-UX require version 5.0.2.)

► One of the following Web browsers on the computer from which you will use the Web Administration Tool. (This might or might not be the computer where the Web Administration Tool is installed.)

- On Windows platforms

  Microsoft Internet Explorer Version 6.0

- On AIX

  Mozilla 1.3 or 1.4

- On xSeries Linux

  Mozilla 1.3 or 1.4

- On iSeries, pSeries, zSeries Linux

  No browser support available

- On Solaris 7, 8, or 9

  Mozilla 1.3 or 1.4

- On HP-UX

  Mozilla 1.3 or 1.4

6.4 Installing the server

Use the information in the following sections to install ITDS 5.2 on AIX using the Installshield GUI.

6.4.1 Create a user ID for ITDS

Before you install, create or be sure that you have created the user ID that will own ITDS's DB2 database used to store the directory data. You will be asked to provide this user ID and its password during configuration, which runs automatically after installation. Keep the following items in mind when creating the user ID:

► The user must have a home directory and must be the owner of the home directory.

► You should create a group called dbsysadm (if it does not already exist). The group ownership of the user's home directory should be that group. For example, in the case of a user named ldapdb2, the user ID home directory should be owned by ldapdb2:dbsysadm.

- ► The user *root* must be a member of the user's primary group (in this case dbsysadm). If root is not a member of this group, add root as a member of the group.

- ► For best results, the user's login shell should be the Korn shell script (`/usr/bin/ksh`).

- ► The user's password must be set correctly and ready to use. For example, the password cannot be expired or waiting for a first-time validation of any kind. (The best way to verify that the password is correctly set is to telnet to the same computer and successfully log in with that user ID and password.)

- ► When configuring the database, it is not necessary, but customary, to specify the home directory of the user ID as the database location. However, if you specify some other location, the user's home directory still must have 3 to 4 MB of space available. This is because DB2 creates links and adds files into the home directory of the instance owner (that is, the user account) even though the database itself is elsewhere. If you do not have enough space in the home directory, you can either create enough space or specify another directory as the home directory.

6.4.2 Installing ITDS with the Installshield GUI

To install:

1. On the computer where you are installing the IBM Tivoli Directory Server, stop any programs that are running and close all windows, if you have any open.

2. If you are installing from a CD, insert the CD in your CD-ROM drive and mount the CD.

3. If you have downloaded a tape archive (tar) file, go to the directory where you extracted the tar file.

4. From the root directory on the CD or the directory where you extracted the tar file, type `./setup`. A language window is displayed.

5. Select the language you want to use during IBM Tivoli Directory Server installation. Click **OK**.

> **Note:** This is the language used in the installation program, not in IBM Tivoli Directory Server. You choose the language used in IBM Tivoli Directory Server in step 10.

6. On the Welcome window, click **Next**.

7. After reading the Software license agreement, select **I accept the terms in the license agreement**. Click **Next**.

8. Any preinstalled components and corresponding version levels are displayed. Click **Next**.

9. To install to the default directory, click **Next**. You can specify a different directory by clicking **Browse**.

> **Note:** Do not use special characters, such as hyphen (-) and period (.) in the name of the installation directory. If you do not use the default location, use a name such as ldap or ldapdir. Do not use a name such as ldap-dir or ldap.dir.

10. Select the language you want to use in IBM Tivoli Directory Server 5.2. Click **Next**.

11. A window showing the following components for installation is displayed, as shown in Figure 6-1 on page 136:

   – Client SDK 5.2
   – Web Administration Tool 5.2
   – Server 5.2
   – IBM WebSphere Application Server - Express 5.0.2
   – DB2 V8.1
   – GSKit

   The components that are not yet installed are preselected. You can choose to reinstall the server, the client, or the Web Administration Tool if they were previously installed.
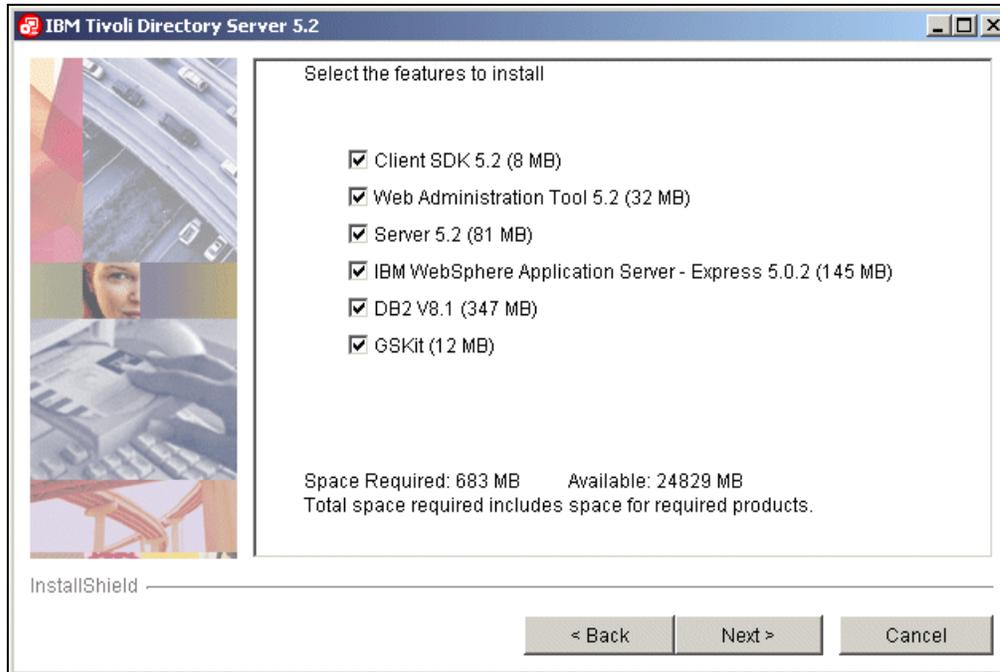
Figure 6-1   Install component selection window

Figure 6-1 also indicates the amount of disk space required and available on the selected drive.

Be sure the components you want to install are selected, and click **Next**.

12. The installation program now has enough information to begin installing. A summary window displays the components you selected and the locations where the selected components will be installed. Click **Back** to change any of your selections. Click **Next** to begin installation.

13. After the files are installed:

   – If you installed the client, the Client Readme file is displayed. Read the file and click **Next**.

   – If you installed the server, the server Readme file is also displayed. Read the file and click **Next**.

   – If you installed the Web Administration Tool, the Web Administration Tool Readme file is also displayed. Read the file and click **Next**.

At this point in the installation, the ITDS Configuration Tool is automatically executed so that you can complete the server configuration. Before you can use

the server, you must set the administrator DN and password and configure the database that will store the directory data.

6.4.3 Configuring the Administrator DN and password

Each ITDS Server has a special "super-user" account associated with it that provides maximum privileges within ITDS. You will need to create this account before you can administer ITDS.

To set the administrator DN and password, refer to Figure 6-2 on page 138 and perform the following steps:

1. In the IBM Tivoli Directory Server Configuration Tool window, click **Administrator DN/password** in the task list on the left.

2. In the Administrator DN/password window on the right, type a valid DN (or accept the default DN, cn=root) in the Administrator DN field.

   The IBM Directory Server administrator DN is the DN used by the administrator of the directory. This administrator is the one user who has full access to all data in the directory.

   The default DN is cn=root. DNs are not case sensitive. If you are unfamiliar with X.500 format, or if for any other reason you do not want to define a new DN, accept the default DN.

3. Type the password for the Administrator DN in the Administrator Password field. You must define a password. Passwords are case-sensitive.

   Record the password for future reference.

   > **Note:** Double byte character set (DBCS) characters in the password are not supported.

4. Retype the password in the Confirm password field.

5. Click **OK**.

Figure 6-2   Setting the Administrator DN and password

6.4.4 Configuring the database

Since ITDS uses IBM DB2 Universal Database as the storage repository for all data, prior to adding data to your directory, you will need to configure a database instance that will be associated with ITDS.

To configure the directory database:

1. Before you configure the database that ITDS will use, create or be sure that you have previously created a valid user ID that will own the DB2 database used to store the directory data. You will be asked to provide this user ID and its password during configuration, which runs automatically after the base installation.

> **Note:** Verify that the user ID you have created or assigned can successfully log into the system. Check to ensure the password does not expire on first login. Check to see if the account is enabled.

2. In the Configuration Tool, click **Configure database** in the task list on the left.

Figure 6-3   Database configuration - Configuring the database

3. Select **Configure New Database** in the left panel and click **Next**, as shown in Figure 6-3.

4. A user ID and password are requested; refer to Figure 6-4 on page 140:

   a. Type a user ID in the User ID field. This user ID must already exist before you can configure the database. This is the user ID you created in step 1. Type a password for the user in the Password field. Passwords are case-sensitive.

   b. Click **Next**.
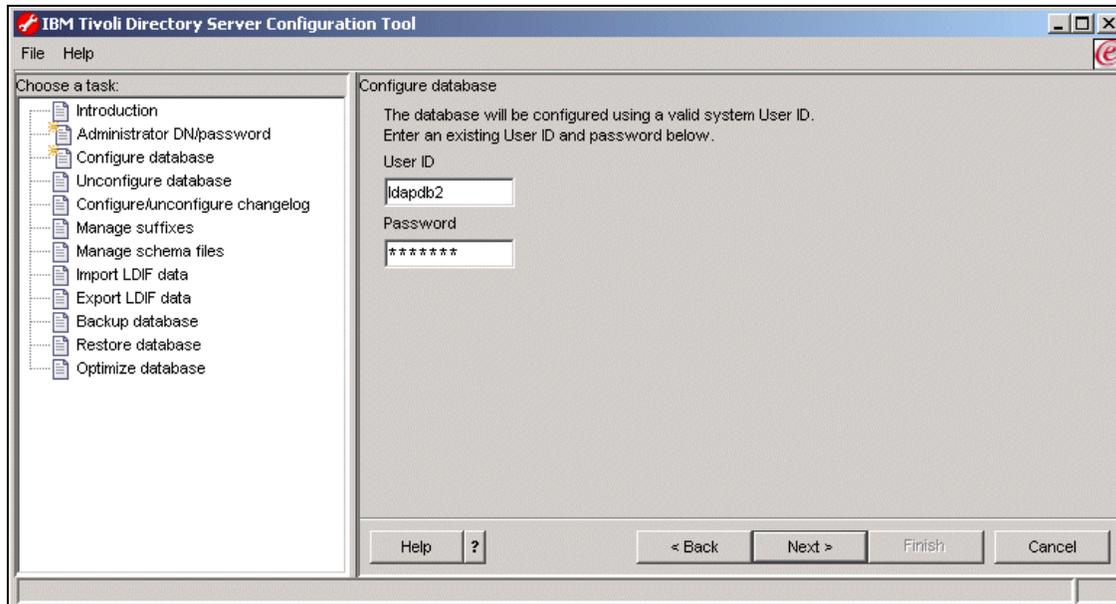
Figure 6-4   Database configuration - Setting the user ID and password for the database

5. Next you will be prompted for a name for the database, as shown in Figure 6-5 on page 141.

   Type the name you want to give the DB2 database. The name can be from 1 to 8 characters long. The database will be created in an instance with the same name as the user ID.

6. Click **Next**.
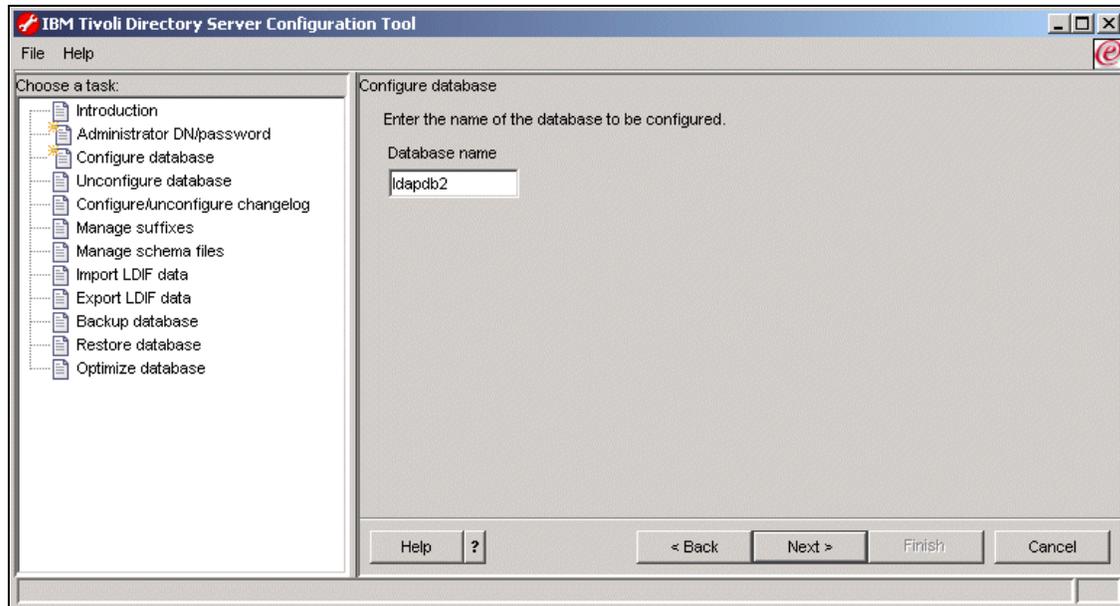
Figure 6-5   Database configuration - Choose DB2 database name

7. If the database location is requested, as shown in Figure 6-6 on page 142:

    a. Type the location for the database in the Database location field. For AIX, this must be a location on the file system, typically the home directory of the user you created earlier in the installation.

        Be sure that you have at least 80 MB of free hard disk space in the location you specify and that additional disk space is available to accommodate growth as new entries are added to the directory.

    b. Click **Next**.
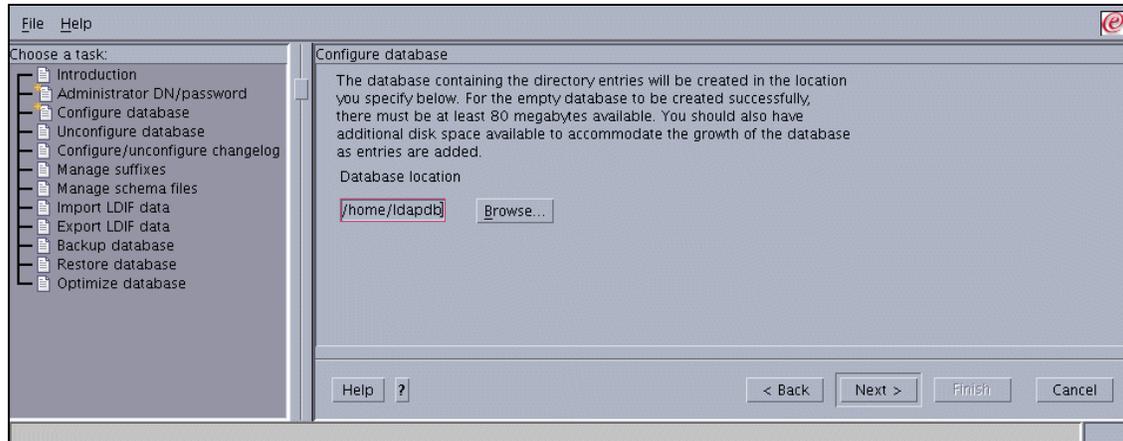
Figure 6-6   Database configuration - Choosing an install location (AIX)

8. If a character set selection is requested, as shown in Figure 6-7 on page 143:

   a. Click the type of database you want to create. You can create a UCS Transformation Format (UTF-8) database, in which LDAP clients can store UTF-8 character data, or a local code page database, which is a database in the local code page.

Note: IBM Tivoli Directory Server supports a wide variety of national language characters through the UTF-8 (UCS Transformation Format) character set. As specified for the LDAP Version 3 protocol, all character data that is passed between an LDAP client and a server is in UTF-8. Consequently, the directory server can be configured to store any national language characters that can be represented in UTF-8. The limitations on what types of characters can be stored and searched for are determined by how the database is created. The database character set can be specified as UTF-8 or it can be set to use the server system's local character set (based on the locale, language, and code page environment).

If you specify UTF-8, you can store any UTF-8 character data in the directory. LDAP clients running anywhere in the world (in any UTF-8 supported language) can access and search the directory. In many cases, however, the client has limited ability to properly display the results retrieved from the directory in a particular language/character set. There is also a performance advantage to using a UTF-8 database because no data conversion is required when storing data to or retrieving data from the database.

    b. Click **Next**.

Figure 6-7   Database configuration - Codepage selection

9. In the verification window shown in Figure 6-8, information is displayed about the configuration options you specified. To return to an earlier window and change information, click **Back**. To begin configuration, click **Finish**.
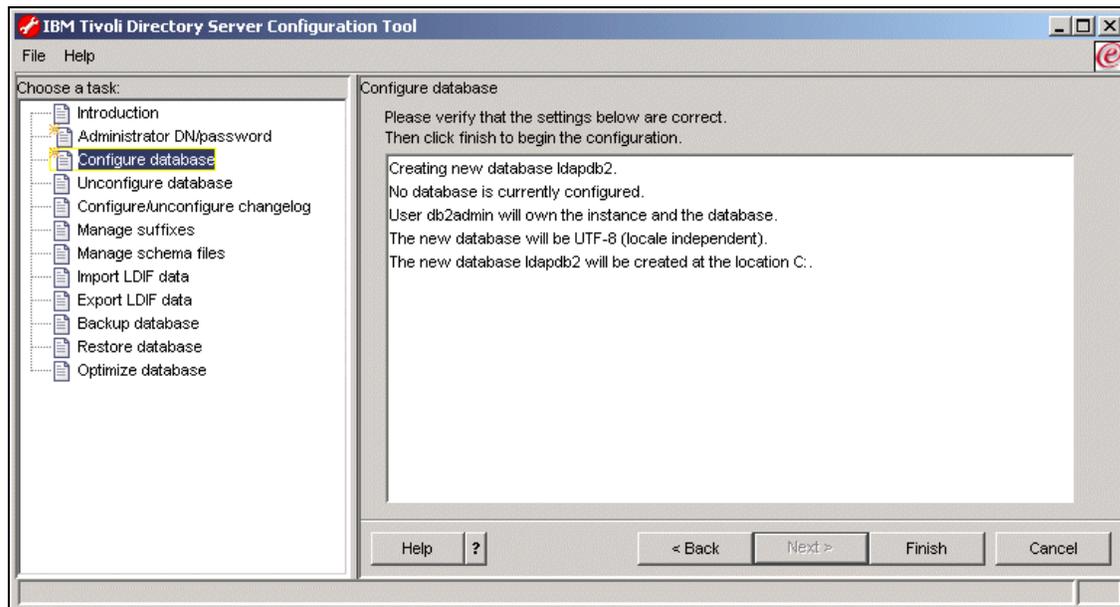
Figure 6-8   Configuration final confirmation

10. The completion window is displayed as shown in Figure 6-9 on page 145. Click **Close**.
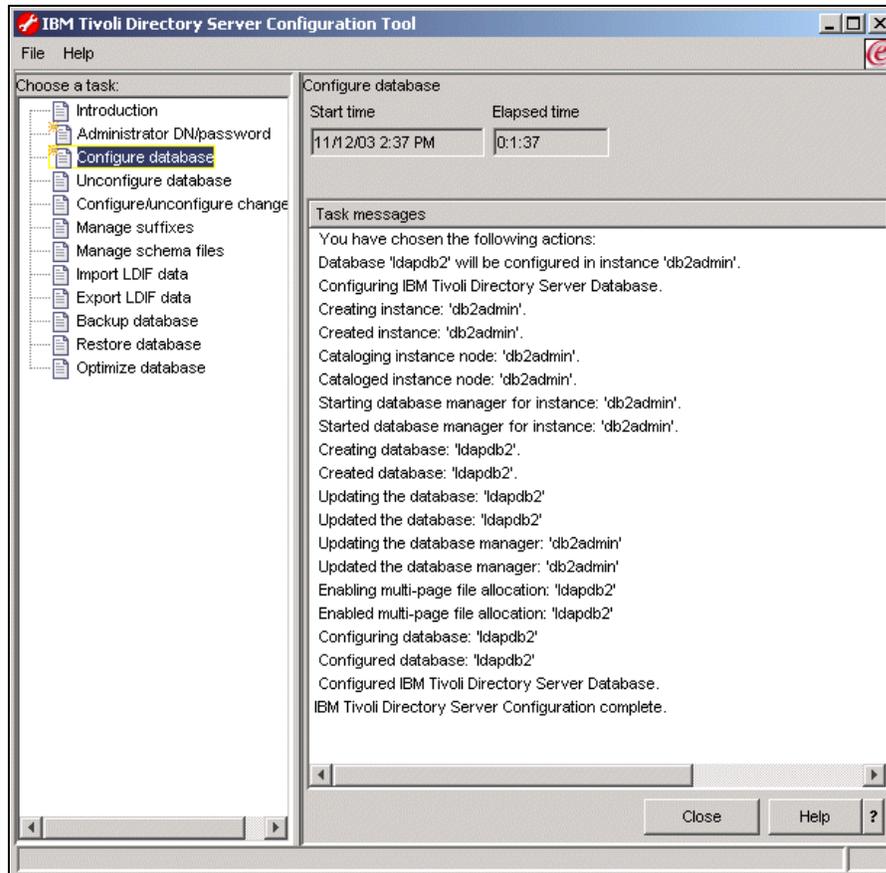
Figure 6-9   Database configuration - Results window

6.4.5 Adding a suffix

A suffix (also known as a naming context) is a distinguished name (DN) that identifies the top entry in a locally held directory hierarchy. Because of the relative naming scheme used in LDAP, this DN is also the suffix of every other entry within that directory hierarchy. A directory server can have multiple suffixes, each identifying a locally held directory hierarchy, for example, `o=ibm,c=us`.

Entries to be added to the directory must have a suffix that matches the DN value, such as `ou=Marketing,o=ibm,c=us`. If a query contains a suffix that does not match any suffix configured for the local database, the query is referred to the LDAP server that is identified by the default referral. If no LDAP default referral is

specified, an `Object does not exist` result is returned. The server must be stopped before you add or remove suffixes.

Add a suffix

Refer to Figure 6-10 and perform the following steps to add a suffix:

1. In the Configuration Tool, click **Manage suffixes** in the task list on the left.

2. In the Manage suffixes window, type the suffix you want to add in the SuffixDN field, and click **Add**.

3. When you have added all the suffixes you want, click **OK**. When you click **Add**, the suffix is added to the list in the Current suffix DNs box; however, the suffix is not actually added to the directory until you click **OK**.
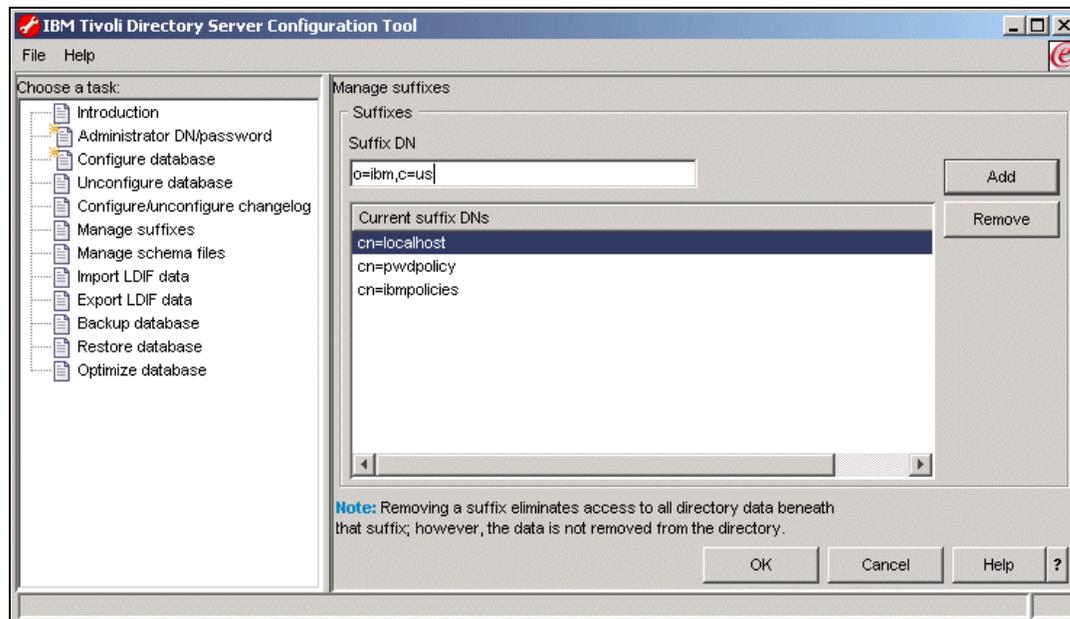
Figure 6-10   Adding a suffix

Removing a suffix

To remove a suffix:

1. In the Configuration Tool, click **Manage suffixes** in the task list on the left.

2. In the Manage suffixes window, click the suffix you want to remove in the Current suffix DNs box, and click **Remove**.

3. When you have selected all the suffixes you want to remove, click **OK**. When you click **Remove**, the suffix is removed from the list in the Current suffix DNs box; however, the suffix is not actually removed until you click **OK**.

6.4.6 Removing or reconfiguring a database

At some point you may need to remove the DB2 database instance that is associated with ITDS. The ITDS `ldapxcfg` tool allows you to *unconfigure* the database instance, *unconifgure* and *destroy* the database instance, and *unconfigure*, *destroy*, and *delete* the database instance.

To unconfigure the database, refer to Figure 6-11 on page 148 and perform the following steps:

1. In the Configuration Tool, click **Unconfigure database** in the task list on the left.

2. In the Unconfigure database window, click of the following:

   – Unconfigure only

   Does not destroy any existing LDAP DB2 data. However, the configuration information for the database will be removed from the configuration file (ibmslapd.conf), and the database will be inaccessible to the directory server.

   – Unconfigure and destroy database

   Removes the existing database and its contents, and removes the configuration information for the database from the configuration file.

   – Unconfigure and destroy database and delete instance

   Removes the existing database and its contents, removes the configuration information for the database from the configuration file, and deletes the instance in which the database is located.

3. Click **Unconfigure**.

Figure 6-11   Unconfiguring the DB2 database associated with ITDS

Once you have completed these steps, you may now configure or re-configure a new database instance for use with ITDS. See "Configuring the database" on page 138 for more information.

6.4.7 Enabling and disabling the change log

The change log database is used to record changes to the schema or directory entries in the typical LDAP entry structure that can be retrieved through the LDAP API. The change log records all update operations: *Add, delete, modify,* and *modrdn*. The change log enables LDAP client applications to retrieve a set of changes that have been made to an IBM Tivoli Directory Server database. The client might then update its own replicated or cached copy of the data.

The change log function causes all updates to LDAP to be recorded in a separate change log DB2 database (that is, a different database from the one used to hold the LDAP server Directory Information Tree). The change log database can be used by other applications to query and track LDAP updates. The change log function is disabled by default.

Unlike some other directory servers on the market, the change log is not required by ITDS for replication to work successfully. Typically, the change log is enabled so meta-directory sychronization products such as IBM Tivoli Directory Integrator (ITDI) can detect changes occurring within ITDS and then push those changes to other non-ITDS data repositories.

There are some performance considerations when you enable the change log since all changes within ITDS are now logged to a separate a database instance. You should evaluate the impact of enabling the change log during in the pre-deployment phases of your ITDS deployment.

You can use the `ldapxcfg` Configuration Tool to enable or disable the change log. The server must be stopped before you enable or disable the change log.

To enable the change log, refer to Figure 6-12 on page 150 and perform the following steps:

1. In the Configuration Tool, click **Configure/unconfigure changelog** in the task list on the left.

2. In the Configure/unconfigure changelog window, select the **Enable change log database** check box.

3. In the Maximum number of log entries box, click **Unlimited** if you want an unlimited number of entries in the change log. If you want to limit the number of entries, click **Entries** and type the maximum number of entries you want recorded. The default is 1,000,000 entries.

4. In the Maximum age box, accept the default of **Unlimited** if you want entries to remain in the change log indefinitely, or click **Age** and type the number of days and hours for which you want each entry to be kept.

5. Click **Update**.

Figure 6-12   Enabling the change log

To disable the change log:

1. In the Configuration Tool, click **Configure/unconfigure changelog** in the task list on the left.

2. In the Configure/unconfigure changelog window, clear the Enable change log database check box.

3. Click **Update**.

6.5 Starting ITDS

There are a number of other optional tasks you can perform within the Directory Configuration tool at this point such as adding custom schema and importing data. Those tasks do not have to be completed before you initially start the server. Those topics are covered in subsequent chapters.

The easiest way to start the server is by typing `ibmslapd` at a AIX command prompt. The output of this command is shown in Example 6-1.

Example 6-1   Starting the Directory Server

```
test_aix:# ibmslapd
Server starting.
Plugin of type EXTENDEDOP is successfully loaded from libevent.so.
Plugin of type EXTENDEDOP is successfully loaded from libtranext.so.
Plugin of type EXTENDEDOP is successfully loaded from libldaprepl.so.
Plugin of type PREOPERATION is successfully loaded from libDSP.so.
Plugin of type PREOPERATION is successfully loaded from libDigest.so.
Plugin of type EXTENDEDOP is successfully loaded from libevent.so.
Plugin of type EXTENDEDOP is successfully loaded from libtranext.so.
Plugin of type AUDIT is successfully loaded from /lib/libldapaudit.so.
Plugin of type EXTENDEDOP is successfully loaded from libevent.so.
Plugin of type EXTENDEDOP is successfully loaded from libtranext.so.
Plugin of type DATABASE is successfully loaded from /lib/libback-rdbm.so.
Plugin of type REPLICATION is successfully loaded from /lib/libldaprepl.so.
Plugin of type EXTENDEDOP is successfully loaded from /lib/libback-rdbm.so.
Plugin of type EXTENDEDOP is successfully loaded from libevent.so.
Plugin of type DATABASE is successfully loaded from /lib/libback-config.so.
Plugin of type EXTENDEDOP is successfully loaded from libloga.so.
Non-SSL port initialized to 389.

test_aix:#
```

After you type `ibmslapd` at the command prompt, a number of messages will be logged to the screen. One of them should say, `IBM Tivoli Directory (SSL) Version 5.2 Server started`.

Note: There are a number of other ways to start ITDS. Please refer to Chapter 9, "IBM Tivoli Directory Server Distributed Administration" on page 193, for more information.

To verify ITDS is indeed running, configured properly, and responding to queries, you can type the following command at Unix command prompt:

`ldapsearch -s base -b "" objectclass=*`

The output of this command is shown in Example 6-2.

Example 6-2   Querying the root DSE

```
# ldapsearch -s base -b "" objectclass=*

namingcontexts=CN=SCHEMA
```

```
namingcontexts=CN=LOCALHOST
namingcontexts=CN=PWDPOLICY
namingcontexts=CN=IBMPOLICIES
namingcontexts=O=IBM,C=US
subschemasubentry=cn=schema
supportedextension=1.3.18.0.2.12.1
supportedextension=1.3.18.0.2.12.3
supportedextension=1.3.18.0.2.12.5
supportedextension=1.3.18.0.2.12.6
supportedextension=1.3.18.0.2.12.15
supportedextension=1.3.18.0.2.12.16
supportedextension=1.3.18.0.2.12.17
supportedextension=1.3.18.0.2.12.19
supportedextension=1.3.18.0.2.12.44
supportedextension=1.3.18.0.2.12.24
supportedextension=1.3.18.0.2.12.22
supportedextension=1.3.18.0.2.12.20
supportedextension=1.3.18.0.2.12.28
supportedextension=1.3.18.0.2.12.30
supportedextension=1.3.18.0.2.12.26
supportedextension=1.3.6.1.4.1.1466.20037
supportedextension=1.3.18.0.2.12.35
supportedextension=1.3.18.0.2.12.40
supportedextension=1.3.18.0.2.12.46
supportedextension=1.3.18.0.2.12.37
supportedcontrol=2.16.840.1.113730.3.4.2
supportedcontrol=1.3.18.0.2.10.5
supportedcontrol=1.2.840.113556.1.4.473
supportedcontrol=1.2.840.113556.1.4.319
supportedcontrol=1.3.6.1.4.1.42.2.27.8.5.1
supportedcontrol=1.2.840.113556.1.4.805
supportedcontrol=2.16.840.1.113730.3.4.18
supportedcontrol=1.3.18.0.2.10.15
supportedcontrol=1.3.18.0.2.10.18
security=none
port=389
supportedsaslmechanisms=CRAM-MD5
supportedsaslmechanisms=DIGEST-MD5
supportedldapversion=2
supportedldapversion=3
ibmdirectoryversion=5.2
ibm-ldapservicename=test_aix
ibm-serverId=3d63f6c0-b48f-1027-92b9-ea0c2fc6cccd
ibm-supportedacimechanisms=1.3.18.0.2.26.3
ibm-supportedacimechanisms=1.3.18.0.2.26.4
ibm-supportedacimechanisms=1.3.18.0.2.26.2
vendorname=International Business Machines (IBM)
vendorversion=5.2
ibm-sslciphers=N/A
```

```
ibm-slapdisconfigurationmode=FALSE
ibm-slapdSizeLimit=500
ibm-slapdTimeLimit=900
ibm-slapdDerefAliases=always
ibm-supportedAuditVersion=2
ibm-sasldigestrealmname=test_aix
```

If the suffix you added in "Adding a suffix" on page 145 is displayed in the output of your `ldapsearch` command in the format `namingcontexts=O=IBM,C=US` (`o=ibm,c=us` is the suffix added in this example), then ITDS's `slapd` LDAP listener is configured properly and open for business.

6.6 Uninstalling ITDS

To uninstall ITDS, issue the following commands:

1. As the operating system user root, kill `ibmslapd` if it is running.

2. Type:

   `su -ldapdb2`

3. Type:

   `cd sqllib`

4. Type:

   `. ./db2profile`

   Note that there is a period<space> in front of the `./db2profile`.

5. Type:

   `db2stop`

6. Type:

   `exit`

7. (Optional) If you want to remove the DB2 Database associated with ITDS, type `ldapucfg -d -r -i` (select **Continue**). If you do not remove the database, it will still be available later on if you re-install the ITDS.

8. Type `/usr/ldap/_uninst/uninstall`. Note that the installer is a X-Windows application and you will need to have a local X-Windows console or have exported your display to another machine that has X-Windows running on it. Follow all the prompts until the uninstallation is complete

The basic uninstallation of ITDS is complete. ITDS does leave files behind in different locations including /opt/IBM/db2, /var/ldap, and /usr/lda.

7

ITDS installation and basic configuration on Intel Linux

This section describes the installation and basic configuration of ITDS 5.2 on Intel Linux based platforms. For the latest information and updates, as well as code downloads, please check the IBM site at:

http://www-3.ibm.com/software/tivoli/products/directory-server/

ITDS 5.2 has several installation options. You can install using an InstallShield graphical user interface (GUI) or use platform-specific installation methods such as the command line or installation tools for the operating system. This chapter focuses on the GUI installation. For more information on the other types of installation options, please refer to the ITDS documentation at:

http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html

Before installing, see *IBM Tivoli Directory Server Version 5.2 Server Readme*, GI11-4151, for any updated information about supported versions of the Linux operating system. The readme file is in the root directory of the CD or the directory where you extracted the server package. After installing, the readme file is located in the *installpath*\doc*lang* directory in files server.txt, server.pdf, and server.htm, where:

► *installpath* is the location where IBM Tivoli Directory Server is installed.

► *lang* is the locale you chose when you installed IBM Tivoli Directory Server; for example, for United States English the locale is en_US.

Also see the *IBM Tivoli Directory Server Version 5.2 Readme Addendum* which contains the latest information. The latest version of the *Readme Addendum* can be found online with the ITDS product documentation:

`http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html`

7.1 Installable components

When you install IBM Tivoli Directory Server, you can install either the client or the server, which requires the client.

In addition, you can install the Web Administration Tool on an application server, with or without the server or the client. You can use the Web Administration Tool to administer IBM Tivoli Directory Server servers either locally or remotely. You can install a single Web Administration console to manage multiple IBM Tivoli Directory Server servers. You can manage servers from previous releases, including SecureWay Directory 3.2.x and IBM Directory Server version 4.1 and 5.1. See Requirements for the Web Administration Tool in "Web Administration Tool" on page 161 for a complete list of servers that can be managed.

► Client: (Required) Includes a number of key libraries and command utilities required by the server. The client also includes a "C" Development SDK. This component can be installed standalone and requires no other components to be installed. GSKit must be installed if you require SSL for stronger security.

► Server: (Required) The core LDAP server component. You must install at least the client and DB2 in conjunction with the server.

► IBM GSKit: (Optional) IBM Global Security Kit (GSKit) Version 7a is an software package that is required only if Secure Sockets Layer (SSL) Security or Transport Layer Security (TLS) is required.

► IBM WebSphere Express Application Server: (Optional) To use the Web Administration Tool, an application server is required. The embedded version of IBM WebSphere Application Server - Express V5.0.2 is provided with ITDS as an application server.

> **Note:** During the writing of this book, the IBM WebSphere Express Application Server did not function properly on Red Hat Enterprise Linux (RHEL) 3. Do not install it until this issue has been resolved.

► Web Administration Tool: (Optional) A Web-based tool used to manage any number of distributed IBM Tivoli Directory Servers as well as prior versions of IBM's Directory Server product line. In order to install the Web Administration tool, you will need to have a supported Application Server already installed or the bundled IBM WebSphere Express Application Server is required.

► IBM DB2: (Required) IBM DB2 Universal Database is used as the underling data storage mechanism for the server.

In order to install the server, at a bare minimum you must install a client, server, and DB2. If you want to require secure access over SSL to the LDAP Server or Web Administration Tool, you will also need to install GSKIT. Finally, if you have

not yet installed the Web Administration Tool anywhere else, you will need to install it along with a supported application server.

7.2 Installation and configuration checklist

Below you will find an abbreviated checklist that contains a high-level summary of the steps required to install and configure ITDS to the point where you can add your own data. Many of these steps are optional but all are recommended in order to provide a well-tuned, high-performance, and secure directory service environment.

ITDS 5.2 installation checklist:

1. Verify that the hardware and operating system meet minimum requirements. See "System and software requirements" on page 159.

2. Obtain product including latest relevant Fixpacks.

3. Operating system configuration and tuning.

4. Basic Product Installation. See "Installing the server" on page 162.

5. Add Administrator DN and password. See "Configuring the Administrator DN and password" on page 166.

6. Configure database. See "Configuring the database" on page 167.

7. Add suffix. See "Adding a suffix" on page 173.

8. Tune DB2. See "DB2 tuning" on page 491.

9. Tune `slapd` parameters in `ibmslapd.conf`. See "Additional slapd and ibmslapd settings" on page 488.

10. Schema customization. See "Modifying the schema" on page 292.

11. Configure ITDS.

    c. TCP/IP Ports ITDS uses.

    d. Password encryption. See "Password encryption" on page 451.

    e. Password policy enforcement. See "Password policy enforcement" on page 437.

    f. SSL / TLS, Kerberos, and Digest-MD5. See "SSL/TLS support" on page 455.

    g. Log locations and settings. See "Enabling and disabling the change log" on page 176

7.3 System and software requirements

To install the IBM Tivoli Directory Server client and server packages, administer the server, and use the IBM Global Security Kit (GSKit), your computer must meet the minimum system requirements as outlined in this section.

7.3.1 ITDS Client

The IBM Tivoli Directory Server Client SDK provides the tools required to develop LDAP applications as well as a number of the most commonly used command line utilities for manipulating LDAP data within the directory. The following are provided:

► Client libraries that provide a set of C-language APIs

► C header files for building and compiling LDAP applications

► Documentation that describes the programming interface and the sample programs

► Sample programs in source form

► Executable versions of the sample programs:

  – `ldapmodrdn`: LDAP modify relative distinguished name
  – `ldapdelete`: LDAP delete
  – `ldapmodify`: LDAP modify
  – `ldapsearch`: LDAP search
  – `ldapadd`: LDAP add (a renamed version of ldapmodify)
  – `ldapchangepwd`: LDAP change password
  – `ldapexop`: LDAP extended operations

The following are the system and software requirements for the ITDS client on Linux.

► Operating system requirements

  – Red Hat Enterprise Linux 3.0
  – UnitedLinux 1.0
  – SuSE Linux Enterprise Server 8

► Memory requirements

  A minimum of 128 MB RAM is required. For better results, use 256 MB or more.

► Disk space requirements

  You have at least 100 MB of free space in the /var directory and at least 200 MB of free space in the /tmp directory.

► Other requirements

The following additional requirements may apply:

– The Korn shell is required.

– To use IBM GSKit, the IBM JRE or JDK 1.4.1 or an equivalent JRE or JDK is required.

7.3.2 ITDS Server (including client)

The server consists of the following components:

► The server executable `ibmslapd`

► Command line import/export utilities

► Web-based GUI for administering the directory: Web Administration Tool

► Server configuration and database utilities GUI for configuring the directory: Configuration Tool (`ldapxcfg`)

► On-line Web Administration Tool and Configuration Tool helps

► The ITDS Client

The requirements are:

► Operating system requirements

– UnitedLinux 1.0 (including SP2®)
– SuSE Linux Enterprise Server 8
– Red Hat Enterprise Linux 3.0

► Memory requirements

A minimum of 256 MB RAM is required. For better results, 512 MB or more is recommended.

► Disk space requirements

– You must have at least 100 MB of free space in the /var directory and at least 400 MB in the /tmp directory.

– You will need 460–660 MB of disk space for the ITDS software on the device you choose to install onto. If DB2 is already installed, then you will need 160 MB to install the other ITDS components.

– Disk space required for data storage is dependent upon the number and size of database entries. Allow a minimum of 80 MB for your database on Linux systems. Also, ensure that there is approximately another 4 MB of disk space in the home directory of the user who will own the database to create the DB2 instance.

▶ Other software

   – The Korn shell is required.

   – IBM DB2 Universal Database for Linux Version 8.1 Enterprise Server Edition with FixPak 2 (DB2) is included with the IBM Tivoli Directory Server, although DB2 Version 7.2 with FixPak 5 or later is also supported.

7.3.3 Web Administration Tool

You can install the Web Administration Tool on a computer with or without the client or the server. The Web Administration Tool can be used to administer LDAP servers of the following types:

▶ IBM Tivoli Directory Server 5.2
▶ IBM Directory Server 5.1
▶ IBM Directory Server 4.1
▶ IBM SecureWay Directory 3.2.2
▶ IBM OS/400 V5R3
▶ IBM z/OS R4

Note that for IBM z/OS R4, only the following setups are supported:

▶ A single TDBM backend
▶ A single SDBM backend
▶ One TDBM and SDBM backend

The Web Administration Tool is supported on the following versions of Linux:

▶ UnitedLinux 1.0
▶ SuSE Linux Enterprise Server 7 or 8
▶ Red Hat Advanced Server 2.1

To use the Web Administration Tool, you also need the following:

▶ One of the following application servers:

   – The embedded version of IBM WebSphere Application Server - Express V5.0 or later. Version 5.0.2 is provided with IBM Tivoli Directory Server 5.2. (iSeries Linux, pSeries Linux, and HP-UX require version 5.0.2.) If you have version 5.0, which was provided with IBM Tivoli Directory Server, installed, see the embedded version of IBM WebSphere Application Server - Express V5.0 or later. Version 5.0.2 is provided with IBM Tivoli Directory Server 5.2. (iSeries Linux, pSeries Linux, and HP-UX require version 5.0.2.) If you have version 5.0, which was provided with IBM Directory Server, installed, see the section titled *Migrating the Web Administration Tool and upgrading the embedded version of WebSphere Application Server - Express* in the *IBM Tivoli Directory Server Installation and Configuration Guide Version 5.2*, SC32-1338.

- IBM WebSphere 5.0 or later. (iSeries Linux, pSeries Linux, and HP-UX require version 5.0.2.)

► One of the following Web browsers on the computer from which you will use the Web Administration Tool. (This might or might not be the computer where the Web Administration Tool is installed.)

- On Windows platforms

  Microsoft Internet Explorer Version 6.0

- On AIX

  Mozilla 1.3 or 1.4

- On xSeries Linux

  Mozilla 1.3 or 1.4

- On iSeries, pSeries, zSeries Linux

  No browser support available

- On Solaris 7, 8, or 9

  Mozilla 1.3 or 1.4

- On HP-UX

  Mozilla 1.3 or 1.4

7.4 Installing the server

Use the information in the following sections to install ITDS 5.2 on Linux using the Installshield GUI.

7.4.1 Create a user ID for ITDS

Before you install, create or be sure that you have created the user ID that will own ITDS's DB2 database used to store the directory data. You will be asked to provide this user ID and its password during configuration, which runs automatically after installation. Keep the following items in mind when creating the user ID:

► The user must have a home directory and must be the owner of the home directory.

► You should create a group called dbsysadm (if it does not already exist). The group ownership of the user's home directory should be that group. For example, in the case of a user named ldapdb2, the user ID home directory should be owned by ldapdb2:dbsysadm.

- ► The user *root* must be a member of the user's primary group (in this case dbsysadm). If root is not a member of this group, add `root` as a member of the group.
- ► For best results, the user's login shell should be the Korn shell script (`/usr/bin/ksh`).
- ► The user's password must be set correctly and ready to use. For example, the password cannot be expired or waiting for a first-time validation of any kind. (The best way to verify that the password is correctly set is to `telnet` to the same computer and successfully log in with that user ID and password.)
- ► When configuring the database, it is not necessary, but customary, to specify the home directory of the user ID as the database location. However, if you specify some other location, the user's home directory still must have 3 to 4 MB of space available. This is because DB2 creates links and adds files into the home directory of the instance owner (that is, the user account) even though the database itself is elsewhere. If you do not have enough space in the home directory, you can either create enough space or specify another directory as the home directory.

> **Tip:** All of these pre-install steps can be achieved using the following commands. It is assumed that no version of ITDS has been installed previously on the server. Run these commands as the user root:
>
> ```
> groupadd dbsysadm
> usermod -G dbysysadm root
> useradd -G dbsysadm -g dbsysadm ldapdb2 -d /home/ldapdb2 -m
> password ldapdb2 (Change the Password to Something Valid)
> ```
>
> At this point verify the login ID and password work. One way to do this is to type:
>
> ```
> ssh 127.0.0.1 -l ldapdb2
> ```
>
> If your password is accepted and you can login, the password is valid for ITDS use.
>
> Type `exit` to return back to the previous shell.
>
> The directory /home/ldapdb2 should now have permissions that look like:
>
> ```
> drwxr-xr-x    5 ldapdb2  dbsysadm      624 Mar 24 16:25 ldapdb2
> ```
>
> All the user ID and group information should now be set correctly for the ITDS installation.

7.4.2 Installing ITDS with the Installshield GUI

To install:

1. On the computer where you are installing the IBM Tivoli Directory Server, stop any programs that are running and close all windows if you have any open.

2. If you are installing from a CD, insert the CD in your CD-ROM drive and mount the CD.

3. If you have downloaded a tape archive (tar) file, go to the directory where you extracted the tar file.

4. From the root directory on the CD or the directory where you extracted the tar file, type `./setup`. A language window is displayed.

5. Select the language you want to use during IBM Tivoli Directory Server installation. Click **OK**.

> **Note:** This is the language used in the installation program, not in IBM Tivoli Directory Server. You choose the language used in IBM Tivoli Directory Server in step 10.

6. On the Welcome window, click **Next**.

7. After reading the Software license agreement, select **I accept the terms in the license agreement**. Click **Next**.

8. Any preinstalled components and corresponding version levels are displayed. Click **Next**.

9. To install to the default directory, click **Next**. You can specify a different directory by clicking **Browse**.

> **Note:** Do not use special characters, such as hyphen (-) and period (.) in the name of the installation directory. If you do not use the default location, use a name such as ldap or ldapdir. Do not use a name such as ldap-dir or ldap.dir.

10. Select the language you want to use in IBM Tivoli Directory Server 5.2. Click **Next**.

11. A window showing the following components for installation is displayed, as shown in Figure 7-1 on page 165:
    – Client SDK 5.2
    – Web Administration Tool 5.2
    – Server 5.2

– IBM WebSphere Application Server - Express 5.0.2
– IBM DB2 V8.1
– IBM GSKit

The components that are not yet installed are preselected. You can choose to reinstall the server, the client, or the Web Administration Tool if they were previously installed.

> **Note:** During the writing of this book, the IBM WebSphere Express Application Server did not function properly on Red Hat Enterprise Linux (RHEL) 3. Do not install it until this issue is resolved.
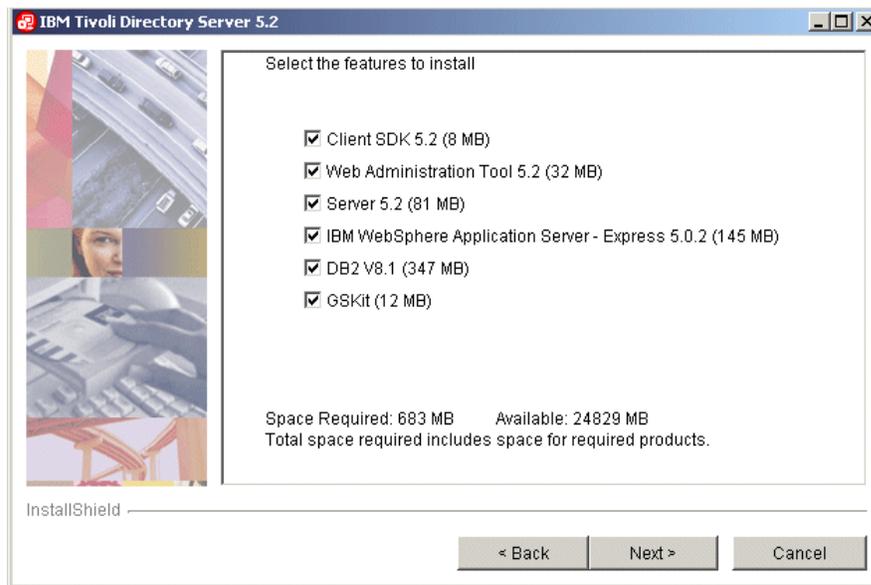
Figure 7-1   Install Component Selection screen

Figure 7-1 also indicates the amount of disk space required and available on the selected drive.

Be sure the components you want to install are selected, and click **Next**.

12. The installation program now has enough information to begin installing. A summary window displays the components you selected and the locations where the selected components will be installed. Click **Back** to change any of your selections. Click **Next** to begin installation.

13. After the files are installed:

– If you installed the client, the Client Readme file is displayed. Read the file and click **Next**.

– If you installed the server, the server Readme file is also displayed. Read the file and click **Next**.

– If you installed the Web Administration Tool, the Web Administration Tool Readme file is also displayed. Read the file and click **Next**.

The ITDS Configuration Tool is automatically executed so that you can complete the server configuration. Before you can use the server, you must set the administrator DN and password and configure the database that will store the directory data.

7.4.3 Configuring the Administrator DN and password

Each ITDS Server has a special "super-user" account associated with it that provides maximum privileges within ITDS. You will need to create this account before you can administer ITDS.

To set the administrator DN and password, refer to Figure 7-2 on page 167 and perform the following steps:

1. In the IBM Tivoli Directory Server Configuration Tool window, click **Administrator DN/password** in the task list on the left.

2. In the Administrator DN/password window on the right, type a valid DN (or accept the default DN, `cn=root`) in the Administrator DN field.

   The IBM Directory Server administrator DN is the DN used by the administrator of the directory. This administrator is the one user who has full access to all data in the directory.

   The default DN is `cn=root`. DNs are not case sensitive. If you are unfamiliar with X.500 format, or if for any other reason you do not want to define a new DN, accept the default DN.

3. Type the password for the Administrator DN in the Administrator Password field. You must define a password. Passwords are case-sensitive.

   Record the password for future reference.

   Note: Double byte character set (DBCS) characters in the password are not supported.

4. Retype the password in the Confirm password field.

5. Click **OK**.

Figure 7-2   Setting the Administrator DN and password

7.4.4 Configuring the database

Since ITDS uses IBM DB2 as the storage repository for all data, prior to adding data to your directory, you will need to configure a database instance that will be associated with ITDS.

To configure the directory database:

1. Before you configure the database that ITDS will use, create or be sure that you have previously created a valid user ID that will own the DB2 database used to store the directory data. You will be asked to provide this user ID and its password during configuration, which runs automatically after the base installation.

> **Note:** Verify that the user ID you have created or assigned can successfully log into the system. Check to ensure the password does not expire on first login. Check to see if the account is enabled.

2. In the Configuration Tool, click **Configure database** in the task list on the left, as shown in Figure 7-3 on page 168.
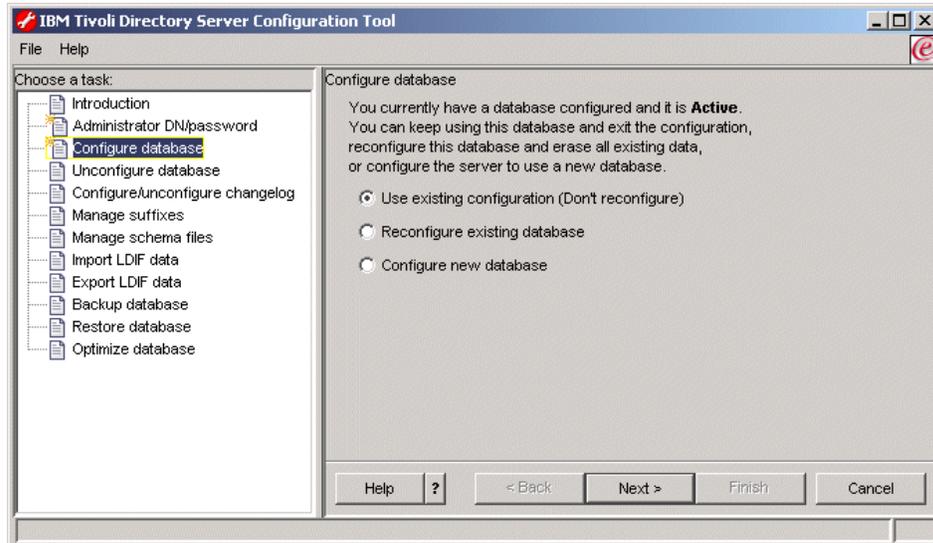
Figure 7-3   Database configuration - Configuring the database

3. Select **Configure New Database** in the left panel and click **Next**.

4. A user ID and password is requested, as shown in Figure 7-4 on page 169:

   a. Type a user ID in the User ID field. This user ID must already exist before
      you can configure the database. This is the user ID you created in step 1.
      Type a password for the user in the Password field. Passwords are
      case-sensitive.

   b. Click **Next**.
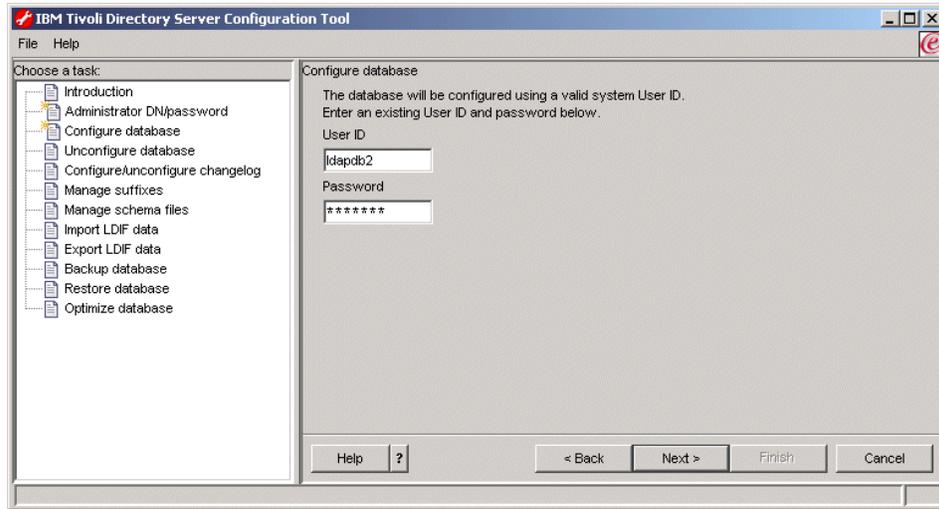
Figure 7-4   Database configuration - Setting the user ID and password for the database

5. Next you will be prompted for a name for the database, as shown in Figure 7-5:

   a. Type the name you want to give the DB2 database. The name can be from 1 to 8 characters long. The database will be created in an instance with the same name as the user ID.

   b. Click **Next**.
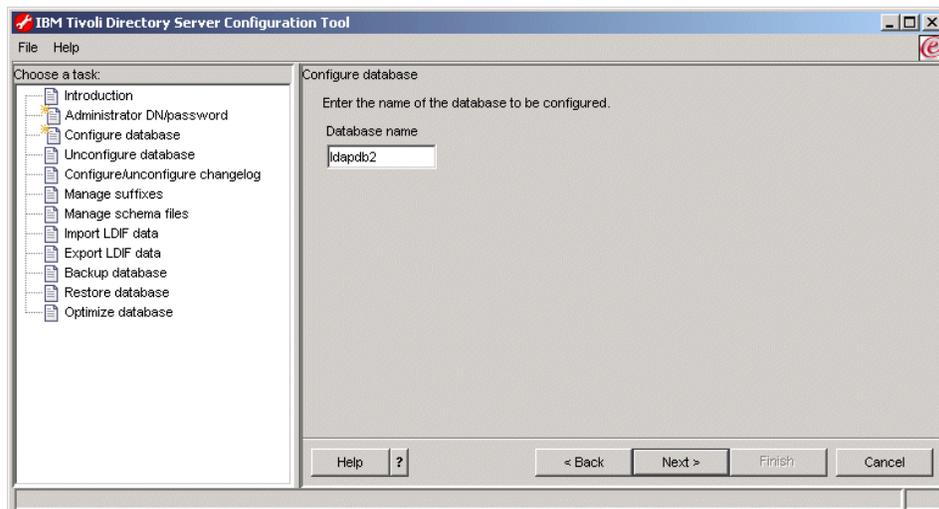
Figure 7-5   Database configuration - Choose DB2 database name

6. If the database location is requested, as shown in Figure 7-6:

    a. Type the location for the database in the Database location field.

        Be sure that you have at least 80 MB of free hard disk space in the location you specify and that additional disk space is available to accommodate growth as new entries are added to the directory.

    b. Click **Next**.
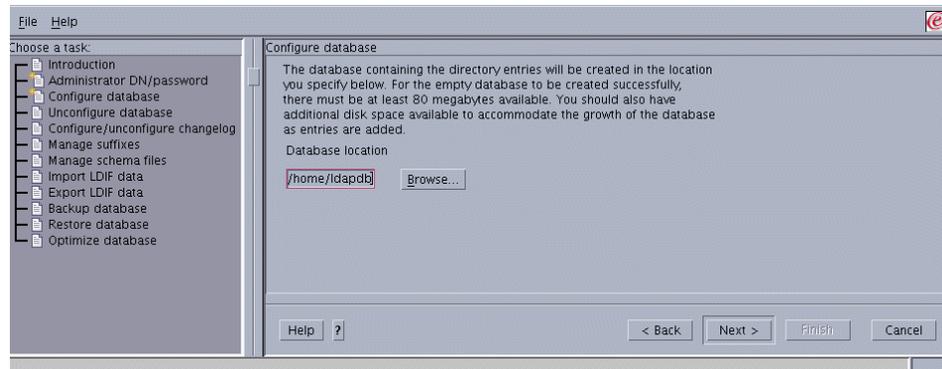
Figure 7-6   Database configuration - Choosing an install locations (Linux)

7. If a character set selection is requested, as shown in Figure 7-7 on page 171:

    a. Click the type of database you want to create. You can create a UCS Transformation Format (UTF-8) database, in which LDAP clients can store UTF-8 character data, or a local code page database, which is a database in the local code page.

> **Note:** IBM Tivoli Directory Server supports a wide variety of national language characters through the UTF-8 (UCS Transformation Format) character set. As specified for the LDAP Version 3 protocol, all character data that is passed between an LDAP client and a server is in UTF-8. Consequently, the directory server can be configured to store any national language characters that can be represented in UTF-8. The limitations on what types of characters can be stored and searched for are determined by how the database is created. The database character set can be specified as UTF-8 or it can be set to use the server system's local character set (based on the locale, language, and code page environment).
>
> If you specify UTF-8, you can store any UTF-8 character data in the directory. LDAP clients running anywhere in the world (in any UTF-8 supported language) can access and search the directory. In many cases, however, the client has limited ability to properly display the results retrieved from the directory in a particular language/character set. There is also a performance advantage to using a UTF-8 database because no data conversion is required when storing data to or retrieving data from the database.

b. Click **Next**.

Figure 7-7   Database configuration - Codepage selection

8. In the verification window shown in Figure 7-8 on page 172, information is displayed about the configuration options you specified. To return to an earlier

window and change information, click **Back**. To begin configuration, click **Finish**.
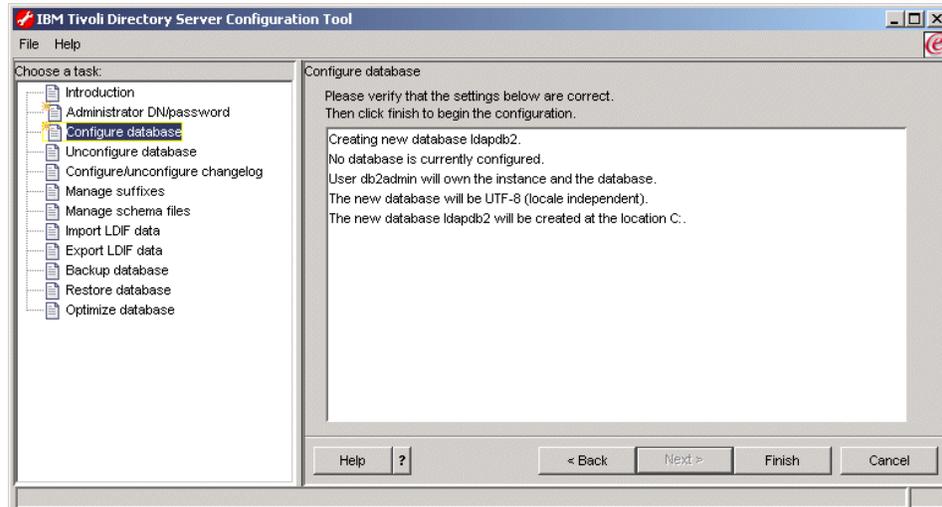
Figure 7-8   Configuration final confirmation

9. The completion window is displayed, as shown in Figure 7-9 on page 173. Click **Close**.
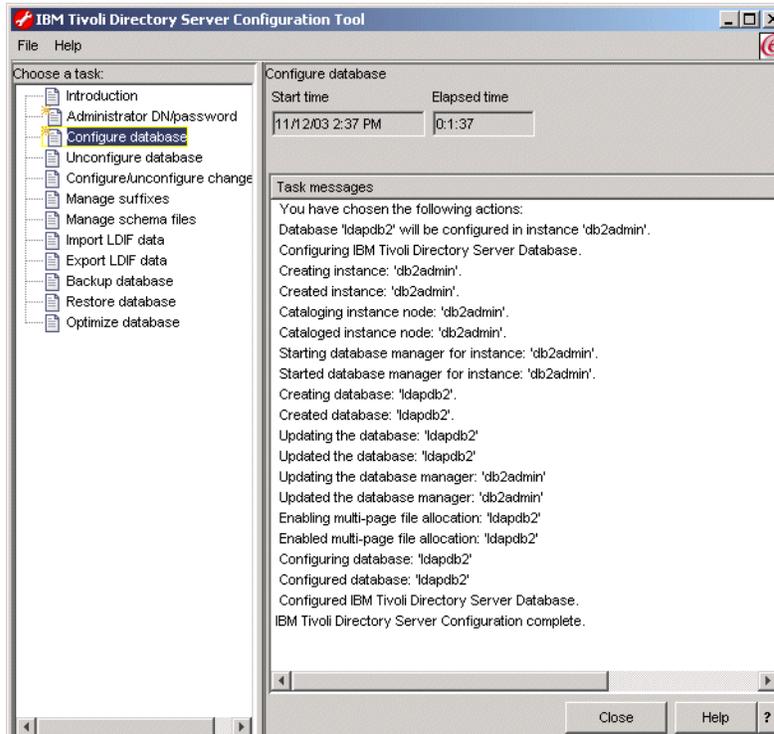
Figure 7-9   Database configuration - Results screen

7.4.5 Adding a suffix

A suffix (also known as a naming context) is a distinguished name (DN) that identifies the top entry in a locally held directory hierarchy. Because of the relative naming scheme used in LDAP, this DN is also the suffix of every other entry within that directory hierarchy. A directory server can have multiple suffixes, each identifying a locally held directory hierarchy, for example, `o=ibm,c=us`.

Entries to be added to the directory must have a suffix that matches the DN value, such as `ou=Marketing,o=ibm,c=us`. If a query contains a suffix that does not match any suffix configured for the local database, the query is referred to the LDAP server that is identified by the default referral. If no LDAP default referral is specified, an `Object does not exist` result is returned. The server must be stopped before you add or remove suffixes.

Add a suffix

To add a suffix refer to Figure 7-10 and perform the following steps:

1. In the Configuration Tool, click **Manage suffixes** in the task list on the left.

2. In the Manage suffixes window, type the suffix you want to add in the SuffixDN field, and click **Add**.

3. When you have added all the suffixes you want, click **OK**. When you click **Add**, the suffix is added to the list in the Current suffix DNs box; however, the suffix is not actually added to the directory until you click **OK**.
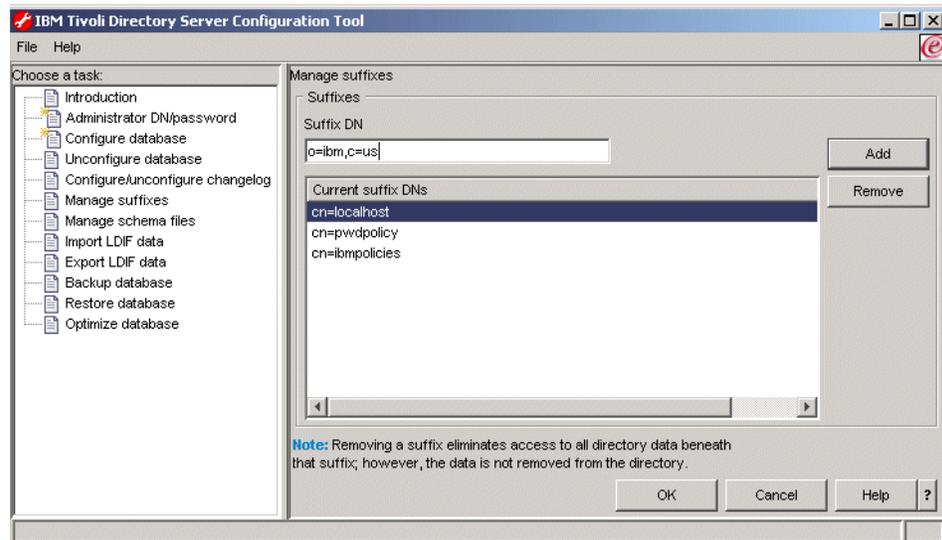
Figure 7-10   Adding a suffix

Removing a suffix

To remove a suffix:

1. In the Configuration Tool, click **Manage suffixes** in the task list on the left.

2. In the Manage suffixes window, click the suffix you want to remove in the Current suffix DNs box, and click **Remove**.

3. When you have selected all the suffixes you want to remove, click **OK**. When you click **Remove**, the suffix is removed from the list in the Current suffix DNs box; however, the suffix is not actually removed until you click **OK**.

7.4.6 Removing or reconfiguring a database

At some point you may need to remove the DB2 database instance that is associated with ITDS. The ITDS `ldapxcfg` tool allows you to unconfigure the

database instance, unconifgure and destroy the database instance, and unconfigure, destroy, and delete the database instance.

To unconfigure the database, refer to Figure 7-11 and perform the following steps:

1. In the Configuration Tool, click **Unconfigure database** in the task list on the left.

2. In the Unconfigure database window, click of the following:

    – Unconfigure only

      Does not destroy any existing LDAP DB2 data. However, the configuration information for the database will be removed from the configuration file (ibmslapd.conf), and the database will be inaccessible to the directory server.

    – Unconfigure and destroy database

      Removes the existing database and its contents, and removes the configuration information for the database from the configuration file.

    – Unconfigure and destroy database and delete instance

      Removes the existing database and its contents, removes the configuration information for the database from the configuration file, and deletes the instance in which the database is located.

3. Click **Unconfigure**.

Figure 7-11   Unconfiguring the DB2 database associated with ITDS

Once you have completed these steps, you may now configure or re-configure a new database instance for use with ITDS. See "Configuring the database" on page 167 for more information.

7.4.7 Enabling and disabling the change log

The change log database is used to record changes to the schema or directory entries in the typical LDAP entry structure that can be retrieved through the LDAP API. The change log records all update operations: Add, delete, modify, and modrdn. The change log enables LDAP client applications to retrieve a set of changes that have been made to an IBM Tivoli Directory Server database. The client might then update its own replicated or cached copy of the data.

The change log function causes all updates to LDAP to be recorded in a separate change log DB2 database (that is, a different database from the one used to hold the LDAP server Directory Information Tree). The change log database can be used by other applications to query and track LDAP updates. The change log function is disabled by default.

Unlike some other directory servers on the market, the change log is not required by ITDS to set up replication. Typically, the change log is enabled so meta-directory sychronization products such as IBM Tivoli Directory Integrator (ITDI) can detect changes occurring within ITDS and then push those changes to other non-ITDS data repositories.

There are some performance considerations when you enable the change log since all changes within ITDS are now logged to a separate a database instance. You should evaluate the impact of enabling the change log during in the pre-deployment phases of your ITDS deployment.

You can use the `ldapxcfg` Configuration Tool to enable or disable the change log. The server must be stopped before you enable or disable the change log.

To enable the change log, refer to Figure 7-12 on page 177 and perform the following steps:

1. In the Configuration Tool, click **Configure/unconfigure changelog** in the task list on the left.

2. In the Configure/unconfigure changelog window, select the E**nable change log database** check box.

3. In the Maximum number of log entries box, click **Unlimited** if you want an unlimited number of entries in the change log. If you want to limit the number of entries, click **Entries** and type the maximum number of entries you want recorded. The default is 1,000,000 entries.

4. In the Maximum age box, accept the default of Unlimited if you want entries to remain in the change log indefinitely, or click **Age** and type the number of days and hours for which you want each entry to be kept.

5. Click **Update**.

Figure 7-12   Enabling the change log

To disable the change log:

1. In the Configuration Tool, click **Configure/unconfigure changelog** in the task list on the left.

2. In the Configure/unconfigure changelog window, clear the Enable change log database check box.

3. Click **Update**.

7.5 Starting ITDS

There are a number of other optional tasks you can perform within the Directory Configuration Tool at this point such as adding custom schema and importing

data. Those tasks do not have to be completed before you initially start the server. Those topics are covered in subsequent chapters.

The easiest way to start the server is by typing `ibmslapd` at a Linux command prompt. The output of this command is shown in Example 7-1.

Example 7-1   Starting the Directory Server

```
test_sles8:# ibmslapd
Server starting.
Plugin of type EXTENDEDOP is successfully loaded from libevent.so.
Plugin of type EXTENDEDOP is successfully loaded from libtranext.so.
Plugin of type EXTENDEDOP is successfully loaded from libldaprepl.so.
Plugin of type PREOPERATION is successfully loaded from libDSP.so.
Plugin of type PREOPERATION is successfully loaded from libDigest.so.
Plugin of type EXTENDEDOP is successfully loaded from libevent.so.
Plugin of type EXTENDEDOP is successfully loaded from libtranext.so.
Plugin of type AUDIT is successfully loaded from /lib/libldapaudit.so.
Plugin of type EXTENDEDOP is successfully loaded from libevent.so.
Plugin of type EXTENDEDOP is successfully loaded from libtranext.so.
Plugin of type DATABASE is successfully loaded from /lib/libback-rdbm.so.
Plugin of type REPLICATION is successfully loaded from /lib/libldaprepl.so.
Plugin of type EXTENDEDOP is successfully loaded from /lib/libback-rdbm.so.
Plugin of type EXTENDEDOP is successfully loaded from libevent.so.
Plugin of type DATABASE is successfully loaded from /lib/libback-config.so.
Plugin of type EXTENDEDOP is successfully loaded from libloga.so.
Non-SSL port initialized to 389.

test_sles8:#
```

After you type `ibmslapd` at the command prompt, a number of messages will be logged to the screen. One of them should say, `IBM Tivoli Directory (SSL) Version 5.2 Server started.`

> **Note:** There are a number of other ways to start ITDS. Please refer to Chapter 9, "IBM Tivoli Directory Server Distributed Administration" on page 193, for more information.

To verify ITDS is indeed running, configured properly, and responding to queries, you can type the following command at the Unix command prompt:

```
ldapsearch -s base -b "" objectclass=*
```

The output of this command is shown in Example 7-2.

Example 7-2   Querying the root DSE

```
# ldapsearch -s base -b "" objectclass=*
```

```
namingcontexts=CN=SCHEMA
namingcontexts=CN=LOCALHOST
namingcontexts=CN=PWDPOLICY
namingcontexts=CN=IBMPOLICIES
namingcontexts=O=IBM,C=US
subschemasubentry=cn=schema
supportedextension=1.3.18.0.2.12.1
supportedextension=1.3.18.0.2.12.3
supportedextension=1.3.18.0.2.12.5
supportedextension=1.3.18.0.2.12.6
supportedextension=1.3.18.0.2.12.15
supportedextension=1.3.18.0.2.12.16
supportedextension=1.3.18.0.2.12.17
supportedextension=1.3.18.0.2.12.19
supportedextension=1.3.18.0.2.12.44
supportedextension=1.3.18.0.2.12.24
supportedextension=1.3.18.0.2.12.22
supportedextension=1.3.18.0.2.12.20
supportedextension=1.3.18.0.2.12.28
supportedextension=1.3.18.0.2.12.30
supportedextension=1.3.18.0.2.12.26
supportedextension=1.3.6.1.4.1.1466.20037
supportedextension=1.3.18.0.2.12.35
supportedextension=1.3.18.0.2.12.40
supportedextension=1.3.18.0.2.12.46
supportedextension=1.3.18.0.2.12.37
supportedcontrol=2.16.840.1.113730.3.4.2
supportedcontrol=1.3.18.0.2.10.5
supportedcontrol=1.2.840.113556.1.4.473
supportedcontrol=1.2.840.113556.1.4.319
supportedcontrol=1.3.6.1.4.1.42.2.27.8.5.1
supportedcontrol=1.2.840.113556.1.4.805
supportedcontrol=2.16.840.1.113730.3.4.18
supportedcontrol=1.3.18.0.2.10.15
supportedcontrol=1.3.18.0.2.10.18
security=none
port=389
supportedsaslmechanisms=CRAM-MD5
supportedsaslmechanisms=DIGEST-MD5
supportedldapversion=2
supportedldapversion=3
ibmdirectoryversion=5.2
ibm-ldapservicename=test_sles8
ibm-serverId=3d63f6c0-b48f-1027-92b9-ea0c2fc6cccd
ibm-supportedacimechanisms=1.3.18.0.2.26.3
ibm-supportedacimechanisms=1.3.18.0.2.26.4
ibm-supportedacimechanisms=1.3.18.0.2.26.2
vendorname=International Business Machines (IBM)
```

```
vendorversion=5.2
ibm-sslciphers=N/A
ibm-slapdisconfigurationmode=FALSE
ibm-slapdSizeLimit=500
ibm-slapdTimeLimit=900
ibm-slapdDerefAliases=always
ibm-supportedAuditVersion=2
ibm-sasldigestrealmname=test_sles8
```

If the suffix you added in "Adding a suffix" on page 173 is displayed in the output of your `ldapsearch` command in the format:

```
namingcontexts=O=IBM,C=US
```

(o=ibm,c=us is the suffix added in this example), then ITDS's `slapd` LDAP listener is configured properly and open for business.

7.6 Quick installation of ITDS 5.2 on Intel (minimal GUI)

If you want to install ITDS quickly and with as little graphical user interface interaction as possible, follow these quick steps:

1. Confirm that the system meets all prerequisites.

2. Log in as the user root and enter the following commands:
   – **groupadd dbsysadm**
   – **usermod -G dbysysadm root**
   – **useradd -G dbsysadm -g dbsysadm ldapdb2 -d /home/ldapdb2 -m**
   – **password ldapdb2** (Change the password to something valid.)

3. At this point verify that the login ID and password work. One way to do this is to type:

   ```
   ssh 127.0.0.1 -l ldapdb2
   ```

   If your password is accepted and you can login the password is valid for IDS use.

4. Type `exit` to return back to the previous shell.

   The directory /home/ldapdb2 should now have permissions that look like:

   ```
   drwxr-xr-x    5 ldapdb2  dbsysadm      624 Mar 24 16:25 ldapdb2
   ```

5. Go to the directory where the setup exists (it may be on a CD-ROM or you may have extracted the tar file into a directory). Type `./setup`. Note that the installer is an X-Windows application and you will need to have a local X-Windows console or have exported your DISPLAY to another machine that has X-Windows running on it.

6. Follow the GUI installer and accept all defaults (pick your local language). For English, the clicks in the GUI you would need to make to get completely through the GUI Install are:

```
OK
NEXT
I ACCEPT
NEXT
ENGLISH
NEXT
NEXT
NEXT
NEXT
NEXT
NEXT
FINISH
```

7. The IBM Tivoli Directory Server Configuration Tool appears. We are not going to use this tool. Exit the tool by clicking:

```
FILE
CLOSE
YES
```

8. Type: `cd /tmp`

9. Type: `ldapcfg -c -a ldapdb2 -w ldapdb3 -d testldap -l /home/ldapdb2` and then select **Continue with the above Actions**. Note that:

   – -c sets the database instance up for UTF-8 storage.

   – -a sets the useraccount that you created.

   – -w sets the password we set for the user that you created.

   – -d sets the name of the DB2 database you want (can be anything).

   – -l sets the directory where the database is created. (Normally this is the home directory of the user that you created.)

The database should configure successfully and return a message similar to:

```
Configuring IBM Tivoli Directory Server Database.
 Creating instance: 'ldapdb2'.
 Created instance: 'ldapdb2'.
 Cataloging instance node: 'ldapdb2'.
 Cataloged instance node: 'ldapdb2'.
 Starting database manager for instance: 'ldapdb2'.
 Started database manager for instance: 'ldapdb2'.
 Creating database: 'testldap'.
 Created database: 'testldap'.
 Updating the database: 'testldap'
 Updated the database: 'testldap'
 Updating the database manager: 'ldapdb2'
```

```
Updated the database manager: 'ldapdb2'
Enabling multi-page file allocation: 'testldap'
Enabled multi-page file allocation: 'testldap'
Configuring database: 'testldap'
Configured database: 'testldap'
Adding local loop back to database: 'testldap'.
Added local loop back to database: 'testldap'.
Stopping database manager for instance: 'ldapdb2'.
Stopped database manager for instance: 'ldapdb2'.
Starting database manager for instance: 'ldapdb2'.
Started database manager for instance: 'ldapdb2'.
Configured IBM Tivoli Directory Server Database.

IBM Tivoli Directory Server Configuration complete.
```

10. Type: `ldapcfg -u"cn=root" -psecret`. Note that:

    – -u sets the Administrator DN.
    – -p sets the Administrator Password.

11. Type: `ldapcfg -s` **"o=ibm,c=us"**. Note that -s sets the suffix you want to use.

12. At this point, configuration is complete. You can type: `ibmslapd` at the command line and the following message should be displayed:

```
Server starting.
Plugin of type EXTENDEDOP is successfully loaded from libevent.so.
Plugin of type EXTENDEDOP is successfully loaded from libtranext.so.
Plugin of type EXTENDEDOP is successfully loaded from libldaprepl.so.
Plugin of type PREOPERATION is successfully loaded from libDSP.so.
Plugin of type PREOPERATION is successfully loaded from libDigest.so.
Plugin of type EXTENDEDOP is successfully loaded from libevent.so.
Plugin of type EXTENDEDOP is successfully loaded from libtranext.so.
Plugin of type AUDIT is successfully loaded from /lib/libldapaudit.so.
Plugin of type EXTENDEDOP is successfully loaded from libevent.so.
Plugin of type EXTENDEDOP is successfully loaded from libtranext.so.
Plugin of type DATABASE is successfully loaded from
/lib/libback-rdbm.so.
Plugin of type REPLICATION is successfully loaded from
/lib/libldaprepl.so.
Plugin of type EXTENDEDOP is successfully loaded from
/lib/libback-rdbm.so.
Plugin of type EXTENDEDOP is successfully loaded from libevent.so.
Plugin of type DATABASE is successfully loaded from
/lib/libback-config.so.
Plugin of type EXTENDEDOP is successfully loaded from libloga.so.
Non-SSL port initialized to 389.
```

13. Basic configuration is complete. Refer to Example 7-2 on page 178 to confirm ITDS is up and running.

7.7 Uninstalling ITDS

To uninstall ITDS, issue the following commands:

1. As the user root, kill ibmslapd if it is running.

2. Type:

   ```
   su -ldapdb2
   ```

3. Type:

   ```
   cd sqllib
   ```

4. Type `. ./db2profile`. (Note: There is a period<space> in front of `./db2profile`.)

5. Type:

   ```
   db2stop
   ```

6. Type:

   ```
   exit
   ```

7. (Optional) If you want to remove the DB2 database associated with ITDS, type: `ldapucfg -d -r -i` (select **Continue**). If you do not remove the database, it will still be available later on if you re-install the ITDS.

8. Type: `/usr/ldap/_uninst/uninstall`. Note that the installer is an X-Windows application and you will need to have a local X-Windows console or have exported your DISPLAY to another machine that has X-Windows running on it. Follow all the prompts until uninstall is complete

The basic uninstallation of ITDS is complete. ITDS does leave files behind in different locations including /opt/IBM/db2, /var/ldap, /usr/ldap/, and other locations. For a more complete uninstall, see "Removing all vestiges of an ITDS 5.2 Install on Intel Linux" on page 183.

7.8 Removing all vestiges of an ITDS 5.2 Install on Intel Linux

The following commands assume you installed the product using the options outlined in "Quick installation of ITDS 5.2 on Intel (minimal GUI)" on page 180.

1. As the user root, kill ibmslapd if it is running.

2. Type: `su -ldapdb2`

3. Type: `cd sqllib`

4. Type: `. ./db2profile` (Note: There is a period<space> in front of the `./db2profile`.)

5. Type: `db2stop`

6. Type: `exit`

7. Type: `ldapucfg -d -r -i` (select **Continue**.)

8. Type: `/usr/ldap/_uninst/uninstall`. Note that the installer is an X-Windows application and you will need to have a local X-Windows console or have exported your DISPLAY to another machine that has X-Windows running on it. Follow all the prompts until uninstall is complete.

9. Type: `cd /tmp`

10. Type: `rm -rf /usr/ldap`

11. Type: `rm -rf /var/ldap`

12. Type: `rm -rf /opt/iBM/db2`

> **Note:** Sometimes IBM WebSphere Express does not uninstall properly. If you see an error indicating it did not uninstall properly, type (as the user root from the command line):
>
> `rpm --erase ldap-webadmind-5.2-1 --justdb`
>
> The version number may vary. Use `yast2` to find out the proper package name and remove it if the version number above is incorrect.

13. Type: `userdel -r ldapdb2`

14. Type: `rm -rf /usr/local/ibm/gsk7`

15. Type: `rm -rf /home/ldapdb2`

16. Type: `groupdel dbsysadm`

At this point, the server should look exactly the way it did before you ever attempted the ITDS install.

8

IBM Tivoli Directory Server installation - IBM zSeries

This chapter provides detailed instructions for installing the IBM Tivoli Directory Server that is packaged with the IBM z/OS operating system. This chapter is based on the IBM z/OS V1R4 operating system. Earlier releases of z/OS may require slight modification of these instructions for proper installation and configuration of the LDAP server.

In this chapter we discuss the following:

► Using the `ldapcnf` utility
► Running the MVS™ jobs generated from the `ldapcnf` utility
► Loading the schema
► Enabling Native Authentication
► Migrating data to LDAP on z/OS

8.1 Installing LDAP on z/OS

The following sections describe the steps needed to install LDAP on the IBM z/OS operating system.

8.1.1 Using the ldapcnf utility

LDAP on z/OS offers a configuration utility called `ldapcnf` to assist in the installation and customization of an LDAP server. To complete the installation process, follow the instructions below.

1. Copy the ldap.profile hfs file from /usr/lpp/ldap/etc to a workable directory such as /etc/ldap.

2. Customize the ldap.profile file to reflect your system and the configuration variables by following the detailed descriptions of each attribute in the profile.

   Note that some attributes in the ldap.profile file are required, but not given a default value. Make sure you read through the entire file, completing all required variables.

3. Run `ldapcnf` from the command line in OMVS. This utility will generate a set of jobs in the MVS dataset that was defined in ldap.profile.

4. Copy the LDAP server started task procedure from the output dataset into the system proclib. The default name for this started task is LDAPSRV.

5. Copy the file named PROGxxx to the system parmlib.

8.1.2 Running the MVS jobs

To do this:

1. Run each job in the following sequence, remembering to check all of the output for successful return codes.

   a. RACF
   b. APF
   c. DBCLI - Make sure DB2 is started before submitting this job.
   d. PGRMCTRL (if required)

2. Use the DB2 SPUFI tool to submit the DBSPUFI job.

3. Start the LDAP server using the LDAPSRV started task. From SDSF you can start the server by entering `/s LDAPSRV`.

4. When you see the phrase `slapd is ready for requests` your LDAP server has started successfully.

8.1.3 Loading the schema

The next steps will assist you in building the LDAP schema and loading the
directory with your suffix and a test user.

1. Copy the following files to your /etc/ldap directory:

   `/usr/lpp/ldap/etc/schema.user.ldif`

   `/usr/lpp/ldap/etc/schema.IBM.ldif`

2. Edit these files (schema.user.ldif and schema.IBM.ldif) by changing the line
   `cn=schema,<suffix>` to reflect the suffix that is defined in your configuration
   file.

3. Use the **ldapmodify** command to load the schema files into the directory.

   ```
   ldapmodify -h x.x.x.x -p 3389 -D "cn=LDAP Administrator" -w secret -f \
   /etc/ldap/schema.user.ldif
   ldapmodify -h x.x.x.x -p 3389 -D "cn=LDAP Administrator" -w secret -f \
   /etc/ldap/schema.IBM.ldif
   ```

4. Create an LDIF file containing the suffix entry for the directory. This may
   contain test users as well. The file may look like the following:

   ```
   dn: o=itso
   objectclass: organization
   objectclass:top
   o: itso

   dn: cn=test1,o=itso
   objectclass: top
   objectclass: ePerson
   cn: test1
   sn: user
   ```

5. Use `ldapadd` to add the entries from the suffix file to the directory.

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

6. Execute the following **ldapsearch** command as an IVP, ensuring that LDAP is
   set up correctly:

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

8.1.4 Enabling Native Authentication

In order to enable LDAP to use a TDBM but bind against RACF, native
authentication must be configured.

1. Copy the following files to your /etc/ldap directory:

   `/usr/lpp/ldap/etc/NativeAuthentication.ldif`

2. Edit the above files by changing the line `cn=schema,<suffix>` to reflect the suffix that is defined in your configuration file.

3. Use the **ldapmodify** command to load the schema files into the directory.

```
ldapmodify -h x.x.x.x -p 3389 -D "cn=LDAP Administrator" -w secret -f \
/etc/ldap/NativeAuthenication.ldif
```

4. Modify the LDAP configuration file to include the following in the TDBM section:

```
useNativeAuth SELECTED
"nativeAuthSubtree" o=itso
nativeUpdateAllowed YES
```

5. Modify existing users, adding the native authentication objectclass and `ibm-nativeId` attribute using the **ldapmodify** command

```
ldapmodify -h x.x.x.x -p 3389 -D "cn=LDAP Administrator" -w secret -f \
/etc/ldap/nativeupdate.ldif
```

nativeupdate.ldif should look like this:

```
dn: cn=test1, o=itso
changetype: modify
add: x
ibm-nativeId: test1
objectclass: ibm-nativeAuthentication
```

8.2 Migrating data to LDAP on z/OS

There are instances where it is necessary to move LDAP data from one platform to another, or simply from one directory to another. This happens when replica servers are being created, or when an LDAP server is being moved to z/OS to take advantage of native authentication.

8.2.1 Migrating LDAP server contents to z/OS

Migrating contents from an existing LDAP server to a z/OS LDAP server can be done using the `DB2LDIF` utility that is packaged with both the z/OS and distributed versions of the IBM Tivoli Directory Server. Examples of using each utility are listed below.

db2ldif on z/OS

`DB2LDIF` is a member of the GLD.GLDSAMP data set and contains JCL for exporting existing LDIF entries from the DB2 database. Export these entries to a temporary file in the file system. LDAP writes the exported file to SYSPRINT. See Example 8-1 on page 189 for an example of this JCL.

Example 8-1   Example DB2LDIF JCL

```
//DB2LDIF JOB (????,????),'AHMADS JOB',MSGCLASS=O,CLASS=A,
//    NOTIFY=????????,REGION=OM,USER=SYSADM1,PASSWORD=SYSADM1
//DB2LDIF PROC REGSIZE=OM,
//    CBCONFIG='/WebSphere390/CB390',
//    PARMS='',
//    GLDHLQ='SYS1.LDAP',
//    OUTCLASS='*',
//    LDAPPATH='etc/ldap',
//    LAPDCONF='bboslapd.conf',
//    SYSPLEX=WSLPLEX,
//    SYSNAME=WSL1
//DB2LDIF  EXEC PGM=GLDDB2LD,REGION=&REGSIZE,
//         PARM=('/&PARMS')
//STEPLIB   DD DSN=&GLDHLQ..SGLDLNK,DISP=SHR
//DSNAOINI DD PATH='&CBCONFIG/&SYSPLEX/&LDAPPATH/&SYSNAME..dsnaoini'
//CONFIG   DD PATH='&CBCONFIG/&SYSPLEX/&LDAPPATH/&SYSNAME..&LAPDCONF'
//SYSPRINT DD  PATH='/u/ahmad/export.ldif'
//CEEDUMP  DD SYSOUT=&OUTCLASS
//SYSERR   DD SYSOUT=&OUTCLASS
//STDOUT   DD SYSOUT=&OUTCLASS
//         PEND
//GO       EXEC DB2LDIF
```

The following command will then load the LDIF file created by the **db2ldif**
command into the z/OS directory.

```
ldapmodify –a –h 127.0.0.1 –p 1389 –D "cn=CBAdmin" –w secret –f \
/u/ahmad/export.ldif
```

8.2.2 Moving RACF users to the TDBM space

Moving RACF user IDs to the TDBM side of the LDAP server seems like a simple
task. However, there is no utility to allow this functionality. As you search against
the SDBM backend and interact with RACF, you will see that if you search for
one particular user, you can retrieve that user's entire record, or filter it to retrieve
only one or two attributes. As you try to extract these fields for more than one
user in a given search, you will see that RACF only returns the fully qualified DNs
that match that search. The specific attributes you requested will not be returned.

As a means to extract the most common RACF attributes to convert each SDBM
user into a TDBM user for use with Native Authentication, a PERL script may be
written to complete nested searches, finding all RACF distinguished names that
match the search criteria, then searching each DN for specific information such
as the RACFID and RACFPROGRAMMERNAME. The script would then be able
to extract those attributes and plug them into a user template, printing them out

to an LDIF file. The LDIF file can then be used to add all users to the TDBM. A sample program and the implementation instructions can be found in Appendix C, "Moving RACF users to TBDM" on page 715.

Part 3

In-depth configuration and tuning

In this part we discuss in-depth configuration and tuning of the IBM Tivoli Directory Server in a distributed environment, client tools available, schema management, group and role management, replication, access control, securing the directory, performance tuning, and how to monitor the IBM Tivoli Directory Server.

9

IBM Tivoli Directory Server Distributed Administration

This chapter provides an overview of the new ITDS 5.x distributed management model including the application server that the Web administration tool runs on. We will also be covering `ibmdiradm`, `ibmslapd`, and management tools like `ibmdirctl`. We will describe how these tools can be used to manage a single server as well as multiple servers.

193

9.1 Web Administration Tool graphical user interface

The IBM Tivoli Directory Server Version 5.2 Web Administration Tool is installed on an application server, such as the embedded version of IBM WebSphere Application Server - Express (WAS) included with the IBM Tivoli Directory Server, and administered through a console. Or you can install it on a existing WebSphere Version 5.0 or later or supported application server. IBM Tivoli Directory Servers that have been added to the console can be managed through the Web Administration Tool without having to have the tool installed on each server. The preferred method of administering the server is by using the Web Administration Tool.

The Web Administration Tool enables an extremely wide range of tasks, such as:

► Basic server administration tasks, including:

– Starting and stopping the server
– Checking server status
– Managing server connections
– Managing connection properties
– Creating, managing, and removing an administrative group
– Creating, managing, and removing unique attributes

► Setting server properties, including:

– Changing server ports and enabling language tags
– Setting performance
– Setting and controlling searches
– Enabling and disabling transaction support
– Enabling and disabling event notification
– Adding and removing suffixes
– Creating and removing referrals

► Configuring security settings, including:

– Configuring TLS and SSL
– Setting the level of encryption
– Setting password encryption
– Setting password policy
– Setting password lockout
– Setting Kerberos
– Setting certificate revocation verification
– Configuring the DIGEST-MD5 mechanism

► Managing the IBM Directory schema, including:

Managing object classes and attributes

- ► Managing replication, including:
  - – Creating and modifying replication topology and replication agreements
  - – Monitoring replication status
- ► Managing logs, including:
  - – Viewing error, DB2, and administration daemon error logs
  - – Modifying the error, DB2, and administration daemon logging settings
  - – Viewing, enabling, and disabling the directory and administration daemon audit logs
- ► Managing directory entries, including:
  - – Browsing the tree
  - – Adding, copying, modifying, and deleting an entry
  - – Managing language tags
  - – Adding or deleting auxiliary object class
  - – Changing group membership
  - – Searching the directory entries with or without filters
- ► Managing Access Control Lists, including performing all functions described in previous sections
- ► Managing group, roles, and proxy authorization group
- ► Performing user-specific tasks, including managing realms, templates, groups, and users

9.2 Starting the Web Administration Tool

To start the Web Administration Tool, you must start the application server in which it was installed. For the embedded version of IBM WebSphere Application Server - Express go to the directory where you installed the IBM Tivoli Directory Server and issue one of the following commands:

- ► For UNIX-based platforms

  ```
  <IDSinstalldir>/ldap/appsrv/bin/startServer.sh server1
  ```

  Note: For Solaris this is:

  ```
  opt/ibmldapc/appsrv/bin/startServer.sh server1
  ```

- ► For Windows-based platforms:

  ```
  <IDSinstalldir>\ldap\appsrv\bin\startServer.bat server1
  ```

> **Note:** If you have other application servers running, ensure that the application server where the Web Administration Tool is installed is not running on the same port as the other application servers.

9.3 Logging on to the console as the console administrator

Before you can start using the Web Administration Tool for the server you want to manage, ensure that you have the completed the following tasks during the configuration of that server:

► You must have set the admin DN and password to be able to start a given server.

► You must have configured a database to be able to start a given server in a state other than configuration only mode.

► You must have the administration daemon running to be able to start, stop, or restart a given server remotely.

To log on as the console administrator, refer to Figure 9-1 on page 197 and perform these steps:

1. Assuming you have installed and started the embedded version of WebSphere Application Server - Express V5.0.2 that ships with ITDS, change your login URL to the following:

   ```
   http://<hostname>:9080/IDSWebApp/IDSjsp/Login.jsp
   ```

   The login page should appear. At the IBM Tivoli Directory Server Web Administration login page log in as Console Admin, the default selection in the LDAP Hostname field.

2. In the Username field type: `superadmin`

3. In the Password field type: `secret`

4. Click **Login**. The IBM Tivoli Directory Server Web Administration Tool console is displayed.
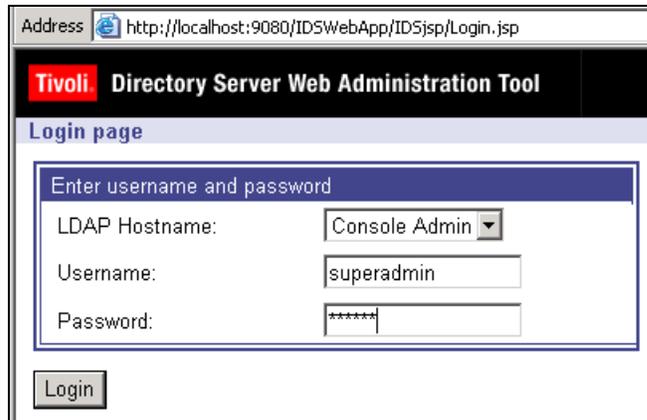
Figure 9-1   Logging in as console administrator

> **Note:** When using the Web Administration Tool, do not open additional login panels from the file options of the browser. Only one instance of the Web Administration can function on a single browser instance. They cannot share the same cookies. Additional login panels must be opened from new instances of the browser.
>
> ► For Unix-based systems:
>
>   Launch new windows from the command line using the & option. For example:
>
>   `mozilla &`
>
> ► For Windows-based systems:
>
>   Internet Explorer - Open additional Internet Explorer windows using the Start window or an Internet Explorer short cut from the desktop.

9.4 Logging on to the console as the server administrator

To log on as the server administrator perform these steps:

1. At the IBM Tivoli Directory Server Web Administration login page select the LDAP host name or IP address for your machine from the drop-down menu.

2. Enter the admin DN and the password for that server (you set these up during the server configuration process).

3. Click **Login**.

The IBM Tivoli Directory Server Web Administration Tool console is displayed with various server management tasks. The server management tasks vary depending upon the capabilities of the server.

> **Note:** The Web Administration Tool does not support logging on to a given server using replication supplier credentials.

9.5 Logging on as member of administrative group or as LDAP user

To log on as a member of the administrative group or an LDAP user, perform these steps:

1. At the IBM Tivoli Directory Server Web Administration login page select the LDAP host name or IP address for your machine from the drop-down menu.

2. Enter the your username (in the form of a DN) and password for that server.

3. Click **Login**.

The IBM Tivoli Directory Server Web Administration Tool console is displayed with various server management tasks. The server management tasks vary depending upon your authority or the capabilities of the server or both.

> **Note:** The Web Administration Tool does not support logging on to a given server using replication supplier credentials.

9.6 Logging off the console

To log off of the console, click **Logout** in the navigation area. The Logout successful panel displays the message:

```
If you have been accidentally logged out then you will need to re-login by
clicking here.
```

Click the word **here** in this message to return to the IBM Tivoli Directory Server Web Administration login page.

9.7 Starting and stopping the server

The server can be started or stopped using either the Web Administration Tool or the command line.

9.7.1 Using Web Administration

> **Note:** The administration daemon (`ibmdiradm`) must be running.

The current status of the server, either started, stopped, or started in configuration mode, is indicated by the icons in the upper left-hand corner of the server status area. The current status is also described in the first sentence of the work area, for example: `The Directory Server is currently running.`

To change the running state of the server, perform these steps:

1. Click **Server Administration** in the Web Administration navigation area and then click **Start/Stop/Restart Server** in the expanded list.

2. The message area displays the current state of the server (stopped, running, or running in configuration only mode). Depending on the state of the server, running or stopped, buttons are enabled for you to change the state of the server as shown in Table 9-1.

    – If the server is running, you can click **Stop** to stop the server, or **Restart** to stop and then start the server.

    – If the server is stopped, you can click **Start** to start the server.

    – Click **Close** to return to the Introduction panel.

Table 9-1   Buttons available based on server status

Server status	Buttons available
Stopped	Start, Close
Running	Stop, Restart, Close
Running in configuration mode	Stop, Restart, Close

3. A message is displayed when the server successfully starts or stops. If you need to perform server configuration maintenance, select the **Start / Restart in configuration only mode** check box. In this mode only the system administrator can bind to the server. All other connections are refused until the server is restarted with the DB2 backends enabled (the Start / Restart in configuration only mode check box deselected).

> **Note:** Configuration maintenance can be done while the server is running.

9.7.2 Using the command line or Windows Services icon

Use the following command to start and stop the server:

```
ibmdirctl [-h <hostname>] [-D <adminDN>] [-w <password>] \
[-p <portnumber>] start|stop|restart|status -- [ibmslapd options]
```

> **Note:** The administration daemon (`ibmdiradm`) must be running.

For Windows systems use the **ibmdirctl** command, or perform the following steps:

1. From the desktop, double-click the My Computer icon.
2. Double-click the Control Panel icon.
3. Double-click the Services icon.
4. To start the server select **IBM Tivoli Directory V5.2** and click **Start**.
5. To stop the server select **IBM Tivoli Directory V5.2** and click **Stop**.

9.8 Console layout

The IBM Tivoli Directory Server Web Administration Tool console consists of five areas:

► Banner area

The banner area located at the top of the panel contains the application name, IBM Tivoli Directory Server Web Administration Tool, and the IBM Logo.

► Navigation area

The navigation area, located on the left side of the panel, displays expandable categories for various console or server tasks. The tasks available vary depending on your authority or the capabilities of the server you are logging onto or both.

► Work area

The work area displays the tasks associated with the selected task in the navigation area. For example, if `Managing server security` is selected in the navigation area, the work area displays the Server Security page and the tabs containing tasks related to setting up server security.

► Server status area

The server status area, located at the top of the work area, indicates the status and the name of the server being administered. It also has two icon links, one to the Start/Stop/Restart procedure and the other to general help

information. When you select a task from the navigation area, the name of the selected task, a link to the error log files, and a link to the task help are also displayed.

> **Note:** If you are logged on as the console administrator, this area displays `Console administrator` and provides an icon link to the table of contents for task helps.

▶ Task status area

The task status area, located beneath the work area, displays the status of the current task.

9.9 Configuration only mode

The IBM Tivoli Directory Server supports LDAP access to the server's configuration settings. An administrator can use the LDAP protocol to query and update the configuration for the server. This feature enables remote administration. For the remote access to be more robust and reliable, the server does not depend on successful initialization of the database backends. It is possible to start the server in configuration only mode with only the `cn=configuration` suffix active. As long as the configuration backend is available, the server starts and accepts LDAP requests. Configuration only mode gives an administrator remote access to the server even when errors are encountered during startup.

The following features are supported in configuration only mode:

▶ Access to the configuration file and log files
▶ Auditing
▶ Event notification
▶ Kerberos
▶ SASL
▶ SSL

The following features are not supported in configuration only mode:

▶ Access to the database
▶ Changelog
▶ Password policy
▶ Replication
▶ Schema changes
▶ Transactions

9.9.1 Minimum requirements for configuration-only mode

The following specify the minimum requirements for configuration-only mode:

► The configuration file must be in the correct LDIF format and the server must be able to locate and read the file.

► The server must be able to read and load the schema according to the configuration file.

► The server must be able to load the configuration plug-in.

9.9.2 Starting LDAP in configuration-only mode

The following methods will start the LDAP server in configuration only mode:

> **Note:** Any failure during server startup will also cause the server to start in configuration only mode.

► Using Web Administration:

Check the Configuration only mode when starting the server through the Web Administration Tool.

► Using the command line:

Specify -a or -A on server startup.

```
ibmslapd -a
```

or

```
ibmdirctl -h <hostname> -D <adminDN> -w <adminpw>-p <portnumber> \
start -- -a
```

9.9.3 Verifying the server is in configuration-only mode

To determine if the server is running in configuration only mode, use one of the following methods.

► Using Web Administration:

If the server has been started in configuration only mode the || icon located between the stop and start icons is highlighted.

► Using the command line:

Issue a search of the root DSE for the attribute ibm-slapdisconfigurationmode. If this attribute is set to true, the server is running in configuration only mode.

```
ldapsearch -s base -b " " objectclass=* ibm-slapdisconfigurationmode
```

9.10 Setting up the console

After you have started the application server, you need to set up the console that is going to manage your directory servers. From the IBM Tivoli Directory Server Web Administration login page, log in as the console administrator and perform the following tasks.

9.10.1 Managing the console

At the IBM Tivoli Directory Server Web Administration Tool console the following tasks can be done to manage the console.

Changing the console administrator login

To change superadmin to a different administrator ID, refer to Figure 9-2 and perform the following steps:

1. Expand Console administration in the navigation area

2. Click **Change console administrator login**.

3. Enter the new administrator ID.

> **Note:** Only one administrator ID is allowed. The superadmin ID is replaced by the new ID that you specified.

4. Enter the current administrator password. The password, *secret,* is the same for the new administrator ID, until you change it.

Figure 9-2   Changing console administrator login

Changing the console administration password

To change the administrator password to another password:

1. Expand Console administration in the navigation area

2. Click **Change console administrator password**.

3. Enter the current password.

4. Enter the new password.

5. Enter the new password again to confirm that there are no typographical errors.

6. Click **OK**.

Adding, modifying, and removing servers in the console

Use the following procedures to add, edit, or delete servers in the console.

Adding a server to the console

To add a server to the console, refer to Figure 9-3 and perform the following steps:

1. Expand Console administration in the navigation area.

2. Click **Manage console servers**. A table for listing of server host names and port numbers is displayed.

3. Click **Add**.

4. Enter the host name address or the IP address of the server. For example:

   `servername.austin.ibm.com`

5. Specify the port numbers or accept the defaults.

6. Specify if the server is SSL enabled.

7. Click **OK** to apply the changes or click **Cancel** to exit the panel without making any changes.
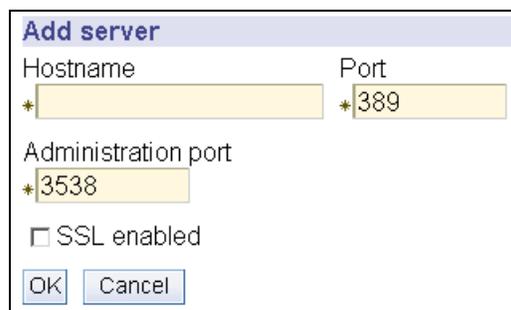
Figure 9-3   Adding a server to the console

Modifying a server in the console

To change the port number or SSL enablement of a server, refer to Figure 9-4 and Figure 9-5, and perform the following steps:

1. Expand Console administration in the navigation area.

2. Click **Manage console servers**. A listing of server host names and port numbers is displayed.

3. Select the radio button next to the server you want to modify.

4. Click **Edit**.

5. You can change the port numbers.

6. You can change whether the server is SSL enabled with the SSL enabled check box.

7. Click **OK** to apply the changes or click **Cancel** to exit the panel without making any changes.
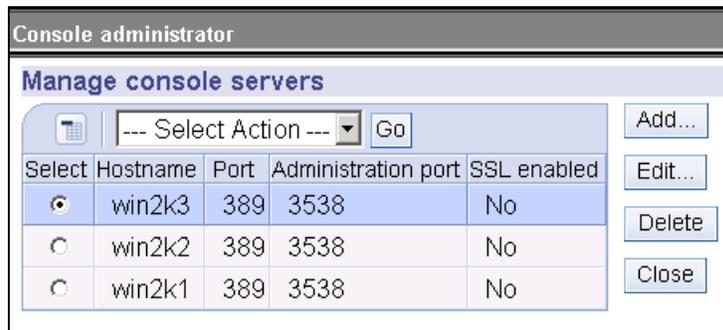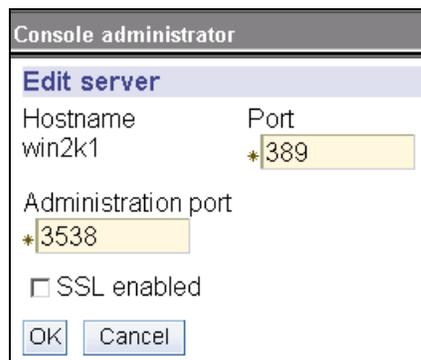

Figure 9-4   Manage console servers

Figure 9-5   Modifying a server in the console

Removing a server from the console

To remove a server from the console, refer to Figure 9-4 on page 205, and perform the following steps:

1. Expand Console administration in the navigation area.

2. Click **Manage console servers**. A listing of server host names and port numbers is displayed.

3. Select the radio button next to the server you want to remove.

4. Click **Delete**.

5. Click **OK** to delete the server or click **Cancel** to exit the panel without making any changes.

Managing console properties

To change the settings for the console properties, perform the following steps:

1. Expand Console administration in the navigation area.

2. Click **Manage console properties**.

3. Click C**omponent management** to specify the components that are enabled for all servers in the console. By default all the components are enabled.

> **Note:** You might not see a management component or some of its tasks, even if it is enabled, if you do not have the correct authority on the server or the server does not have the needed capabilities, or both.

4. Click **Session properties** to set the time-out limit for the console session. The default setting is 60 minutes.

> **Note:** A session might be valid for three to five minutes more than what you have set. This is because the invalidations are performed by a background thread in the application server that acts on a timer interval. This timer interval extends the session time out duration.

5. Click **SSL key database** to set up the console so that it can communicate with other LDAP servers using the Secure Sockets Layer (SSL), if necessary. Set the key database path and file name, the key password, the trusted database path and file name, the trusted password in the appropriate fields. The supported file type is jks.

6. When you have finished setting up the console, click **Logout** to exit.

Component management

Component management allows you to enable or disable management components across all servers in the console. By default all the components are enabled. The components managed from this panel are:

► User properties
► Server administration
► Schema management
► Directory management
► Replication management
► Realms and templates
► Users and groups

Figure 9-6 shows the component management panel. To enable a component, select the check box next to the component. To disable a component, clear the check box next to it.

> **Note:** An enabled management component, or some of the tasks associated with the enabled management component, might not be accessible to a user if one of the following conditions is true:
>
> ► The LDAP server the user is logging into does not support the capabilities required by the management component.
>
> ► The user does not have sufficient access rights on the LDAP server.
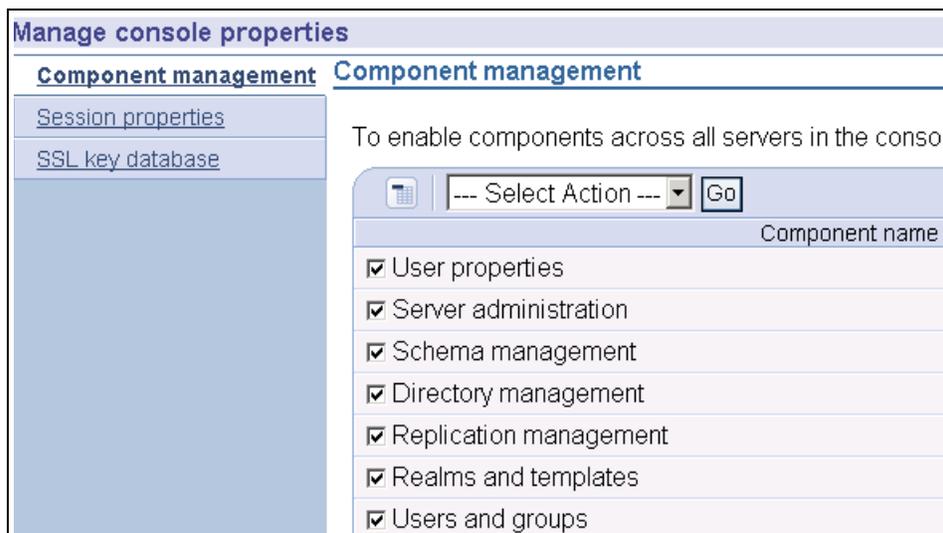

Figure 9-6   Manage console properties

9.10.2 Creating an administrative group

An administrative group provides administrative capabilities without having to share a single ID and password among the administrators. Members of the administrative group have their own unique IDs and passwords. The administrative group member DNs must not match each other and they must also not match the IBM Tivoli Directory Server administrator's DN. Conversely, the IBM Tivoli Directory Server administrator DN must not match the DN of any administrative group member. This rule also applies to the Kerberos or Digest-MD5 IDs of the IBM Tivoli Directory Server administrator and the administrative group members. These DNs must not match any of the IBM Tivoli Directory Server's replication supplier DNs. This also means that IBM Tivoli Directory Server's replication supplier DNs must not match any of the administrative group member DNs or the IBM Tivoli Directory Server administrator DN.

> **Note:** The IBM Tivoli Directory Server's replication supplier DNs can match each other.

The members of the administrative group have all the capabilities of the directory administrator with the following exceptions:

- ► Only the IBM Tivoli Directory Server administrator has the ability to add or remove members from the administrative group. In addition only the IBM Tivoli Directory Server administrator can modify the DN, password, Kerberos ID, or Digest-MD5 ID of any administrative group member. However, a member of the administrative group can modify his own password, but cannot modify his own DN, Kerberos ID, or Digest-MD5 ID. An administrative group member cannot see the password of any other administrative group member or the IBM Tivoli Directory Server administrator.

- ► Only the IBM Tivoli Directory Server administrator has the ability to add or remove the `cn=Keberos,cn=Configuration` and the `cn=Digest,cn=Configuration` entries in the configuration backend. Administrative group members can modify all the attributes in these entries except for the directory administrator's Keberos ID and Digest-MD5 ID.

- ► Only the IBM Tivoli Directory Server administrator has the ability to modify or update any of the audit log settings. Members of the administrative group are able only to view the audit log and the audit log settings.

- ► Only the IBM Tivoli Directory Server administrator has the ability to clear the audit log.

9.10.3 Enabling and disabling the administrative group

Enabling and disabling the administrative group can be done through the Web administration tool and the command line. You must be the IBM Tivoli Directory Server administrator to perform this operation.

> **Note:** In this task and the Manage administrative group tasks that follow, the operation buttons are disabled for members of the administrative group. Members of the administrative group can only view the Administrative group members table on the Manage administrative group panel.

Using Web Administration

To enable or disable the administrative group using the Web Administration Tool, perform the following steps:

1. Expand the Server administration category in the navigation area. Click **Manage administrative group**.

2. To enable or disable the administrative group, click the check box next to Enable administrative group. If the box is checked, the administrative group is enabled.

3. Click **OK**.

> **Note:** If you disable the administrative group, any member who is logged in can continue administrative operations until that member is required to rebind. To stop any additional operations by already bound administrative group members, perform an unbind operation.

Using the command line

To perform the same operations using the command line, issue the following command:

```
ldapmodify -D <adminDN> -w<adminPW> -i<filename>
```

Where the file used is similar to Example 9-1.

Example 9-1   File used for administrative group modification

```
dn: cn=Configuration
cn: Configuration
changetype: modify
replace: ibm-slapdAdminGroupEnabled
#specify TRUE to enable or FALSE to disable the administrative group
#TRUE has been preselected for you.
ibm-slapdAdminGroupEnabled: TRUE
objectclass: top
```

```
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdTop
```

To update the settings dynamically, issue the following **ldapexop** command:

```
ldapexop -D cn=root -w root -op readconfig -scope single \
cn=Configuration ibm-slapdAdminGroupEnabled
```

9.10.4 Adding members to the administrative group

Members can be added to the administrative group through either the Web
Administration Tool or the command line. You must be the IBM Tivoli Directory
Server administrator to perform this operation.

Using Web Administration

To add a member to the administrative group, perform the following steps:

1. On the Manage administrative group panel, click **Add**.

2. On the Add administrative group member panel, enter the member's
   administrator DN (this must be a valid DN syntax).

3. Enter the member's password.

4. Enter the member's password again to confirm it.

5. Optionally, enter the member's Kerberos ID. The Kerberos ID must be in
   either ibm-kn or ibm-KerberosName format. The values are case insensitive,
   for example, ibm-kn=root@TEST.AUSTIN.IBM.COM is equivalent to
   ibm-kn=ROOT@TEST.AUSTIN.IBM.COM.

   Note: This field is only available for the AIX and Windows NT and Windows
   2000 platforms. It is displayed only if the Kerberos supported capabilities
   OID (1.3.18.0.2.32.30) is found on the server.

6. Optionally, enter the member's Digest-MD5 user name.

7. Click **OK**.

   Note: The Digest-MD5 user name is case sensitive. Repeat this procedure
   for each member you want to add to the administrative group.

The member administrator DN, Digest-MD5 username, if specified, and
Kerberos ID, if specified, are displayed in the Administrative group members list
box.

> **Note:** Kerberos support is only available for the AIX and Windows NT, Windows 2000, and Windows 2003 platforms. The Kerberos ID column in the is displayed in the Administrative group members list box only, if the kerberos supported capabilities OID (1.3.18.0.2.32.30) is found on the server.

Using the command line

To perform the same operations using the command line, issue the following command:

```
ldapadd -D <adminDN> -w<adminPW> -i<filename>
```

Where the file used is similar to Example 9-2.

Example 9-2   File used to add user to administrative group

```
dn: cn=AdminGroup, cn=Configuration
cn: AdminGroup
objectclass: top
objectclass: container
dn: cn=admin1, cn=AdminGroup, cn=Configuration
cn: admin1
ibm-slapdAdminDN: <memberDN>
ibm-slapdAdminPW: <password>
#ibm-slapdKrbAdminDN and ibm-slapdDigestAdminUser are optional attributes.
ibm-slapdKrbAdminDN: <KerberosID>
ibm-slapdDigestAdminUser: <DigestID>
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdAdminGroupMember
```

> **Note:** If you already have a member created in the administrative group, omit the first entry in Example 9-2.

To update the settings dynamically, issue the following **ldapexop** command:

```
ldapexop -D cn=root -w root -op readconfig -scope subtree \
cn=AdminGroup,cn=Configuration
```

9.10.5 Modifying an administrative group member

Modifying an administrative group member can be done through either the Web Administration Tool or the command line. You must be the IBM Tivoli Directory Server administrator to perform this operation.

Using Web Administration

To modify an administrative group member's information, on the Manage administrative group panel:

1. Select the member whose information you want to modify.

2. Click **Edit**.

3. Enter the member's administrator DN (this must be a valid DN syntax).

4. Change the member's password.

5. Enter the member's password again to confirm it.

6. Enter or change the member's Kerberos ID. The Kerberos ID must be in either ibm-kn or ibm-KerberosName format. The values are case insensitive, for example, ibm-kn=root@TEST.AUSTIN.IBM.COM is equivalent to ibm-kn=ROOT@TEST.AUSTIN.IBM.COM.

   Note: This field is only available for the AIX and Windows NT and Windows 2000 platforms. It is displayed only, if the Kerberos supported capabilities OID(1.3.18.0.2.32.30) is found on the server.

7. Enter or change the member's Digest-MD5 user name. The Digest-MD5 user name is case sensitive.

8. Click **OK**.

   Note: If you are member of the administrative group, you can change your password using the **User properties** → **Change password** panel.

Repeat this procedure for each member you want to modify in the administrative group.

Using the command line

To perform the same operations using the command line, issue the following command:

```
ldapmodify -D <adminDN> -w<adminPW> -i<filename>
```

Where the file used is similar to Example 9-3.

Example 9-3   File used to modify an administrative group member

```
dn: cn=admin1, cn=AdminGroup, cn=Configuration
cn: admin1
changetype: modify
replace: ibm-slapdAdminDN
ibm-slapdAdminDN: cn=<memberDN>
```

```
        -
replace: ibm-slapdAdminPW
ibm-slapdAdminPW: <password>
        -
replace: ibm-slapdKrbAdminDN
ibm-slapdKrbAdminDN: <KerberosID>
        -
replace: ibm-slapdDigestAdminUser
ibm-slapdDigestAdminUser: <DigestID>
```

To update the settings dynamically, issue the following **ldapexop** command:

```
ldapexop -D cn=root -w root -op readconfig -scope subtree \
cn=AdminGroup,cn=Configuration
```

9.10.6 Removing a member from the administrative group

Removing a member from the administrative group can be done through the
Web administration tool or the command line. You must be the IBM Tivoli
Directory Server administrator to perform this operation.

Using server administration

To remove a member of the administrative group, on the Manage administrative
group panel:

1. Select the member you want to remove.

2. Click **Delete**.

3. You are prompted to confirm the removal.

4. Click **OK** to delete the member or **Cancel** to return to the Manage
   administrative group panel without making any changes.

Repeat this procedure for each member you want to remove from the
administrative group.

Using the command line

To perform the same operations using the command line, issue the following
command:

```
ldapdelete -D <adminDN> -w<adminPW> -i<filename>
```

Where the file used is similar to Example 9-4.

Example 9-4   File used to remove a member of the administrative group

```
#list additional DNs here, one per line
```

```
dn: cn=admin1, cn=AdminGroup, cn=Configuration
```

To remove multiple members, list the DNs. Each DN must be on a separate line.

To update the settings dynamically, issue the following **ldapexop** command:

```
ldapexop -D cn=root -w root -op readconfig -scope subtree \
cn=AdminGroup,cn=Configuration
```

9.11 ibmslapd command parameters

The **ibmslapd** command has two parameters on UNIX systems and an additional two parameters on Windows systems. The following parameters are common to both platforms:

-h *<debug_mask>*

This causes **ibmslapd** to generate debug output to stdout. The *debug_mask* is a bit mask that controls which output is generated with values up to 65535. This parameter is for use by IBM service personnel. See "Server debug mode" on page 214 for more information on the use of this parameter.

-f *<path_to_configuration_file>*

This specifies the location of the configuration file used when starting the server. This parameter is used if you want to use a customized configuration file. If not specified, **ibmslapd** defaults to the platform dependent location where the configuration file was installed.

Additional parameters for Windows systems are:

► **-i *<servicename>***

    This installs IBM Directory as service on the server.

► **-u *<servicename>***

    This removes IBM Directory as service from the server.

Server debug mode

If the error logs do not provide enough information to resolve a problem, you can run the IBM Tivoli Directory Server in a special debug mode that generates very detailed information. The server executable ibmslapd must be run from a command prompt to enable debug output. Be careful not to run in debug mode for long periods of time. This will generate a large amount of data and could easy fill up the max file size of 2 Gb. When this happens, the LDAP will not accept any new connections or log any more data to the file. The way to fix this would be to stop and re-start ibmslapd. When doing this make sure you copy or rename the

trace file before you restart `ibmslapd`. If you do not rename it or copy it, then it will be erased and overwritten with the new trace file.

The syntax is as follows:

```
ldtrc on
ibmslapd -h bitmask
```

Where the specified bitmask value determines which categories of debug output are generated, as shown in Table 9-2.

For example, specifying a bitmask value of 65535 turns on full debug output and generates the most complete information. When you are finished, issue the following command at a command prompt:

```
ldtrc off
```

It is recommended that you contact IBM Service for assistance with interpreting the debug output and resolving of the problem.

Table 9-2   ibmslapd bitmask values and descriptions

Hex	Decimal	Value	Description
0x0001	1	LDAP_DEBUG_TRACE	Entry and exit from routines
0x0002	2	LDAP_DEBUG_PACKETS	Packet activity
0x0004	4	LDAP_DEBUG_ARGS	Data arguments from requests
0x0008	8	LDAP_DEBUG_CONNS	Connection activity
0x0010	16	LDAP_DEBUG_BER	Encoding and decoding of data
0x0020	32	LDAP_DEBUG_FILTER	Search filters
0x0040	64	LDAP_DEBUG_MESSAGE	Messaging subsystem activities and events
0x0080	128	LDAP_DEBUG_ACL	Access Control List activities
0x0100	256	LDAP_DEBUG_STATS	Operational statistics
0x0200	*512*	LDAP_DEBUG_THREAD	Threading statistics
0x0400	1024	LDAP_DEBUG_REPL	Replication statistics
0x0800	2048	LDAP_DEBUG_PARSE	Parsing activities

Hex	Decimal	Value	Description
0x1000	4096	LDAP_DEBUG_PERFORMANCE	Relational backend performance statistics
0x2000	8192	LDAP_DEBUG_RDBM	Relational backend activities (RDBM)
0x4000	16384	LDAP_DEBUG_REFERRAL	Referral activities
0x8000	32768	LDAP_DEBUG_ERROR	Error conditions
0xffff	65535	LDAP_DEBUG_ANY	All levels of debug

9.12 Directory administration daemon

The directory administration daemon (`ibmdiradm`) enables remote management of the IBM Tivoli Directory Server. It must be installed on the machine where the IBM Tivoli Directory Server is installed and must be running continuously. The directory administration daemon accepts requests by way of LDAP extended operations and supports starting, stopping, restarting, and status monitoring of the IBM Tivoli Directory Server. By default, the IBM Directory administration daemon listens on two ports, port 3538 for non-SSL connections and port 3539 for SSL connections, if SSL communication is enabled.

9.12.1 The ibmdiradm command

To start the administration daemon, use the `ibmdiradm` command.

Synopsis

```
ibmdiradm [-h debug_mask] [-f path_to_configuration_file] \
[-s ssl_port] [-p nonssl_port] [-i servicename | -u servicename]
```

Description

Starts the administration daemon.

Options

The options are:

▶ -h *debug_mask*

Causes `ibmdiradm` to generate administration daemon debug output to stdout. The *debug_mask* is a bit mask that controls which output is generated with values up to 65535. This parameter is for use by IBM service personnel.

See "Server debug mode" on page 214 for additional information on debug levels.

► -f *path_to_configuration_file*

Specifies the location of the configuration file used when starting the administration daemon server. This parameter is used if you want to use a customized configuration file. If not specified, `ibmdiradm` defaults to the platform-dependent location where the configuration file was installed.

► -s *ssl_port*

Specifies the SSL port.

► -p *nonssl_port*

Specifies the non-SSL port.

The following two parameters are for Windows systems only:

► -i *servicename*

Adds the administration daemon as a Windows service.

► -u *servicename*

Removes the administration daemon as a Windows service.

Stopping the administration daemon

For UNIX-based systems, run the following commands:

```
ps -ef |grep ibmdiradm
kill -p pid_obtained_by_previous_commnand
```

For Windows systems:

1. Through the Control Panel, open the Services window.

2. Click **Directory Admin Daemon**.

3. Click **Action → Stop**.

9.12.2 Starting the directory administration daemon

Note: By default, the administration daemon is running when you install the IBM Tivoli Directory Server.

For UNIX-based and Windows-based systems issue the command:

```
ibmdiradm
```

For Windows-based systems the directory administration daemon can be started from the control panel (**Control Panel** → **Services**, select **IBM Directory Admin Daemon**, click **Start**).

> **Note:** If you enable SSL communication, the directory administration daemon must be stopped and restarted for SSL to take effect.

9.12.3 Stopping the directory administration daemon

If you have already configured a directory administration DN and password, you can use the **ibmdirctl** command to stop the administration daemon. This command is not platform specific.

```
ibmdirictl -D <adminDN> -w <adminPW> admstop
```

For UNIX-based systems the directory administration daemon can also be stopped by:

```
ps -ef | grep ibmdiradm
kill -p <pid obtained by previous command>
```

For Windows-based systems the directory administration daemon can also be stopped through the control panel (**Control Panel** → **Services**, select **IBM Directory Admin Daemon**, click **Stop**).

9.12.4 Administration daemon error log

The admin daemon error log logs messages pertaining to the ITDS 52 administration daemon. `ibmdiradm` is a lightweight version of `ibmslapd`, which would be needed in case you need to control the server remotely. `ibmdiradm` runs as a daemon on the server and helps remote clients to pass on the start/stop/restart requests to the server. It listens on port 3538 by default for non-SSL communications and over port 3539 by default for the SSL communications. On Windows, the administration daemon is also installed as a service, in addition to the command line version of `ibmdiradm`. The name of the service is *IBM Tivoli Directory Admin Daemon V5.2*. The basic use of `ibmdiradm` is that it is a prerequisite service/daemon for a remote Web Administration GUI to communicate with the server (`ibmslapd`). If the Web Administration GUI is local to ibmslapd, then there is no necessity of `ibmdiradm` to be running for the GUI to communicate to the server.

To control the `ibmdiradm` you would need to use another command-line utility, known as `ibmdirctl`. The details of `ibmdirctl` will be provided in the next section (Figure 9.13 on page 227), but for now refer to Figure 9-7 on page 219 to see which utility controls the other.
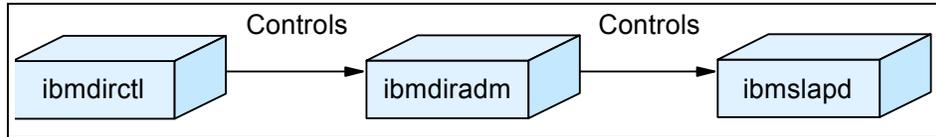
Figure 9-7 Figure depicting the processes for regulating ibmdiradm & ibmslapd

The next question would be why do we need the administration log at all? Well, the answer is quite simple. There are occasions when you need to control your server remotely. There may be issues associated with the remote handling of server, like the `ibmdiradm` is not able to start `ibmslapd`. The administration daemon log is handy in such situations, as we can get to know the probable causes of failure. The problem may be that the server is not configured properly, due to which the server is compelled to start in configuration mode. In such a situation `ibmdiradm` would flash an error saying that it was not able to start the server. You may check out the administration daemon log for getting the relevant details and consequently fix the problem.

Modifying administration daemon error log settings

There are two ways to update the administration daemon error log settings.

Using the Web Administration

Refer to Figure 9-8 on page 220, and perform the following steps:

1. Expand **Logs** in the navigation area, click **Modify admin daemon log settings**.

2. Enter the path and file name for the administration daemon error log. Typically this is the `ibmdiradm.log` file located in the var/ldap/ directory. Ensure that the file exists on the LDAP server and that the path is valid.

> **Note:** var/ldap/ibmdiradm.log is the default administration daemon error log for UNIX systems and installpath\var\ibmdiradm.log is the default administration daemon error log for Windows systems.

3. Click **OK** to apply your changes or click **Cancel** to return to the IBM Tivoli Directory Server Web Administration Welcome panel without making any changes.

4. If you click **OK**, a message is displayed to remind you that you need to restart the server. Click **OK** to return to the IBM Tivoli Directory Server Web Administration Welcome panel.

5. You must stop the server for changes to take effect. After stopping the server you must also stop and start the administration daemon locally to resynchronize the ports.

   For UNIX systems:

   ```
   ibmdirctl -D <AdminDN> -w <Adminpw> admstop
   ibmdiradm
   ```

   For Windows systems:

   – If you are running `ibmdiradm` as a service:

     Through the Control Panel, open the Services window.

     i. Click **Directory Admin Daemon**.
     ii. Click **Action** → **Stop**.
     iii. Click **Directory Admin Daemon**.
     iv. Click **Action** → **Start**.

   – If you are running `ibmdiradm` as a separate process, you just need to kill the current process of `ibmdiradm` and run it again.
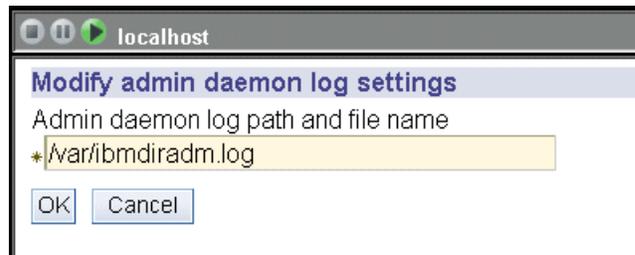
   Restart the server.

Figure 9-8   Settings for the admin daemon log

Using the command line

Issue the command:

```
ldapmodify -D <adminDN> -w >adminPW> -i <filename>
```

Where <filename> contains:

```
dn: cn=Admin, cn=Configuration
changetype: modify
replace: ibm-slapdErrorLog
ibm-slapdErrorLog: <newpathname>
```

You must stop the server for changes to take effect. After stopping the server you must also stop and start the administration daemon locally to resynchronize the

ports. Start the server. The sequence of the commands to do the same is as follows:

```
ibmdirctl -D <adminDN> -w <adminPW> -p 389 stop
ibmdirctl -D <adminDN> -w <adminPW> admstop
ibmdiradm
ibmdirctl -D <adminDN> -w <adminPW> start
```

Viewing the administration daemon error log

Use the following procedures to view the administration daemon error log.

Using Web Administration

Refer to Figure 9-9, and perform the following steps:

1. Expand **Logs** in the navigation area, then click **View admin daemon log**.

2. The panel displays the first page of the administration daemon log and the navigation arrows at the bottom of the panel enable you to go to the next page or to the previous page. From the menu, you can select a specific page, for example Page 6 of 16, and click **Go** to display that page of the administration daemon log.

3. From the Web administration tool you can also:

   a. Click **Refresh** to update the entries in the log.

   b. Click **Clear log** to delete all entries in the administration daemon log.

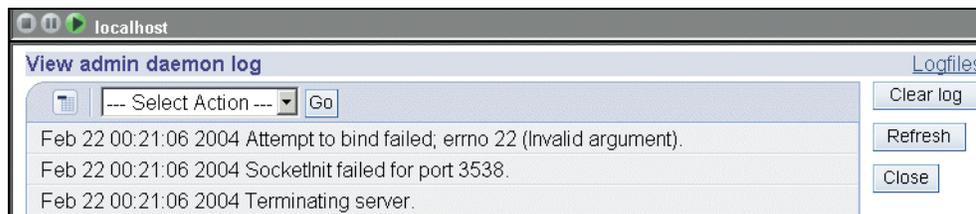   c. Click **Close** to return to the IBM Tivoli Directory Server Web Administration Welcome panel.

Figure 9-9   Contents of the admin daemon log

Using the command line

To view the administration daemon error log issue the following command:

```
more /var/ldap/ibmdiradm.log
```

Where /var/ldap/ibmdiradm.log is your administration daemon error log.

> **Note:** /var/ldap/ibmdiradm.log is the default administration daemon error log
> for UNIX systems and installpath\var\ibmdiradm.log is the default
> administration daemon error log for Windows systems.

Dynamically view and clear administration daemon error log

To dynamically view and clear the administration daemon error log, the following
commands can be used from the command line. See Example 9-5 and
Example 9-6 for sample output from these commands.

```
ldapexop -D <adminDN> -w <adminPW> -op readlog -log ibmdiradm -lines all
ldapexop -D <adminDN> -w <adminPW> -op clearlog -log ibmdiradm
```

Example 9-5   ldapexop command viewing the log

```
E:\>ldapexop -D cn=root -w secret -op readlog -log ibmdiradm -lines all
Feb 22 00:21:06 2004  Attempt to bind failed; errno 22 (Invalid argument).
Feb 22 00:21:06 2004  SocketInit failed for port 3538.
Feb 22 00:21:06 2004  Terminating server.
Mar 03 20:39:11 2004  Open of SSL key database file F:\Keys\server.kdb failed.
Mar 03 20:39:11 2004  Terminating server.
Mar 04 08:46:56 2004  Open of SSL key database file F:\Keys\server.kdb failed.
Mar 04 08:46:56 2004  Terminating server.
Mar 04 09:01:11 2004  Open of SSL key database file F:\Keys\server.kdb failed.
Mar 04 09:01:11 2004  Terminating server.
Mar 04 09:05:58 2004  Open of SSL key database file F:\Keys\server.kdb failed.
Mar 04 09:05:58 2004  Terminating server.
Mar 04 09:08:10 2004  Open of SSL key database file F:\Keys\server.kdb failed.
Mar 04 09:08:10 2004  Terminating server.
Mar 04 09:08:40 2004  Open of SSL key database file F:\Keys\server.kdb failed.
Mar 04 09:08:40 2004  Terminating server.
```

As seen in the example above, there were issues with the key database file,
which was specified during SSL configuration. Hence `ibmslapd` could not be
started by `ibmdiradm`. Consequently the logs of `ibmdiradm` are populated.

Example 9-6   ldapexop command clearing the log

```
E:\>ldapexop -D cn=root -w secret -op clearlog -log ibmdiradm
ibmdiradm log file cleared.
E:\>ldapexop -D cn=root -w secret -op readlog -log ibmdiradm -lines all
Mar 18 08:18:51 2004 Log file cleared.
```

Administration daemon audit logging

We can audit the operations or transactions that are performed between clients
and the directory server via the administration daemon, `ibmdiradm`. This has

again the same uses as we had seen for audit log. We can get a detailed set of timestamped activities occurring on the server. Timestamps obviously play a vital role during problem determination.

> **Note:** Members of the administrative group can view the administration daemon audit log and settings but not modify them. Only the root administrator is enabled to access. Change or clear the administration daemon audit log files.

Administration daemon audit log and administration audit log

To enable the administration daemon audit log you can use the we administration tool, or the command line.

Using Web Administration

Refer to Figure 9-10 on page 224, and perform the following steps to enable the administration audit log and modify the administration audit log settings:

1. Expand **Logs** in the navigation area, click **Modify admin daemon audit log settings**.

2. Select **Enable admin daemon audit logging** to use the audit log utility with the administration daemon.

> **Note:** The default setting is enabled. You only need to select the check box if you have previously disabled the administration daemon audit log.

3. Enter the path and file name for the administration daemon audit log. Typically this is the adminAudit.log file located in the /var/ldap/ directory. Ensure that the file exists on the ldap server and that the path is valid.

> **Note:** /var/ldap/adminAudit.log is the default administration daemon audit log for UNIX systems and installpath\var\adminAudit.log is the default administration daemon audit log for Windows systems.

4. Click **OK** to apply your changes or click **Cancel** to return to the IBM Tivoli Directory Server Web Administration Welcome panel without making any changes.

5. If you click **OK**, a message is displayed to remind you that you need to restart the server. Click **OK** to return to the IBM Tivoli Directory Server Web Administration Welcome panel.

6. You must stop the server for changes to take effect. After stopping the server you must also stop and start the administration daemon locally to resynchronize the ports.

    For UNIX systems:

    ```
    ibmdirctl -D <AdminDN> -w <Adminpw> admstop
    ibmdiradm
    ```

    For Windows systems:

    – If you are running ibmdiradm as a service:

    Through the Control Panel, open the Services window.

    i.   Click **Directory Admin Daemon**.
    ii. Click **Action → Stop**.
    iii. Click **Directory Admin Daemon**.
    iv. Click **Action → Start**.

    – If you are running `ibmdiradm` as a separate process, you just need to kill the current process of `ibmdiradm` and run it again.
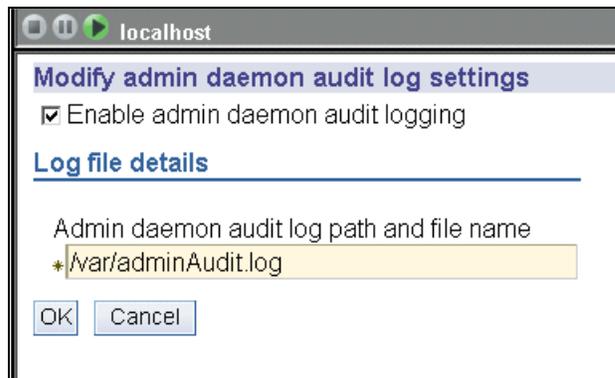
    Restart the server.

Figure 9-10   Settings for the admin daemon audit log

Using the command line

Issue the command:

```
ldapmodify -D <adminDN> -w <adminPW> -i <filename>
```

Where <filename> contains:

```
dn: cn=Admin Audit, cn=Configuration
changetype: modify
replace: ibm-audit
ibm-audit: true
-
```

```
replace: ibm-auditLog
ibm-auditLog: <newpathname>
```

You must stop the server for changes to take effect. After stopping the server you must also stop and start the administration daemon locally to resynchronize the ports. Restart the server.

```
ibmdirctl -D <AdminDN> -w <adminPW> -p 389 stop
ibmdirctl -D <AdminDN> -w <adminPW> admstop
ibmdiradm
ibmdirctl -D <AdminDN> -w <adminPW> start
```

Disabling the administration daemon audit log

To disable audit logging perform the steps in one of the following methods:

Using Web Administration

To use this:

1. Expand **Logs** in the navigation area, click **Modify admin daemon audit log settings**.

2. Deselect **Enable admin daemon audit logging**.

3. Click **OK** to apply your changes or click **Cancel** to return to the IBM Tivoli Directory Server Web Administration Welcome panel without making any changes.

The panel where you would be making these settings is same as the one shown in Figure 9-10 on page 224.

Using the command line

Issue the command:

```
ldapmodify -D <adminDN> -w <adminPW> -i <filename>
```

Where <filename> contains:

```
cn=Admin Audit, cn=Configuration
changetype: modify
replace: ibm-audit
ibm-audit: false
```

> **Note:** If you are using administration daemon audit logging in configuration-only mode, the DN specified is dn: cn=audit, cn=configuration. Any changes made to this DN are overwritten with the dn: cn=audit, cn=localhost values when the server is started in normal mode.

Viewing the administration daemon audit log

Use one of the following procedures to view the administration daemon audit log.

Using Web Administration

Refer to Figure 9-11, and perform the following steps:

1. Expand **Logs** in the navigation area, then click **View admin daemon audit log**.

2. The panel displays the first page of the administration daemon audit log and the navigation arrows at the bottom of the panel enable you to go to the next page or to the previous page. From the menu, you can select a specific page, for example Page 6 of 16, and click **Go** to display that page of the administration daemon audit log.

3. From the Web administration tool, you can:

   a. Click **Refresh** to update the entries in the log.

   b. Click **Clear log** to delete all entries in the administration daemon audit log.

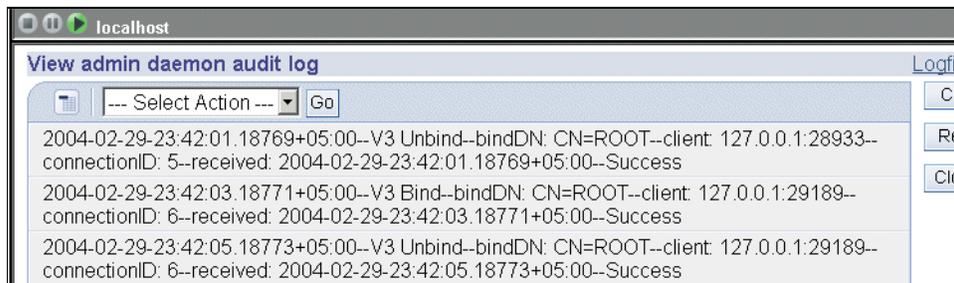   c. Click **Close** to return to the IBM Tivoli Directory Server Web Administration Welcome panel.

Figure 9-11   Contents of the admin daemon audit log

Using the command line

To view the administration daemon audit log issue the following command:

```
more /var/ldap/adminAudit.log
```

Where var/ldap/adminAudit.log is your administration daemon log.

> **Note:** /var/ldap/adminAudit.log is the default administration daemon log for UNIX systems and installpath\var\adminAudit.log is the default administration daemon log for Windows systems.

To dynamically view and clear the administration daemon audit log, the following commands can be used from the command line. See Example 9-7 and Example 9-8 for sample output from these commands.

```
ldapexop -D <adminDN> -w <adminPW> -op readlog -log adminAudit -lines all
ldapexop -D <adminDN> -w <adminPW> -op clearlog -log adminAudit
```

Example 9-7   ldapexop command to view the administration audit log

```
E:\>ldapexop -D cn=root -w secret -op readlog -log adminAudit -lines all | head
-5
2004-03-03-06:29:54.55220+05:00--V3 Bind--bindDN: CN=ROOT--client:
127.0.0.1:34056--connectionID: 12--received:
2004-03-03-06:29:54.55220+05:00--Success
2004-03-03-06:29:54.55220+05:00--V3 Unbind--bindDN: CN=ROOT--client:
127.0.0.1:34056--connectionID: 12--received:
2004-03-03-06:29:54.55220+05:00--Success
2004-03-03-06:29:59.55225+05:00--V3 Bind--bindDN: CN=ROOT--client:
127.0.0.1:34312--connectionID: 13--received:
2004-03-03-06:29:59.55225+05:00--Success
2004-03-03-15:09:03.076+05:00--Audit logging started.
2004-03-04-03:16:53.43747+05:00--Audit logging started.
2004-03-04-03:31:08.44602+05:00--Audit logging started.
```

As seen in Example 9-7, the messages logged appear the same as they were for the error log. This log is, however, pertaining to transactions with `ibmslapd` through `ibmdiradm`.

Example 9-8   ldapexop command to clear the administration audit log

```
E:\>ldapexop -D cn=root -w secret -op clearlog -log adminAudit
adminAudit log file cleared.

E:\>ldapexop -D cn=root -w secret -op readlog -log adminAudit -lines all
Mar 18 08:25:20 2004 Log file cleared.
```

9.13 The ibmdirctl command

This is the administration daemon control program. The administration daemon (`ibmdiradm`) must be running.

> **Note:** Only the administrator may use this utility.

Syntax

The syntax is:

```
ibmdirctl [-D adminDN] [-h hostname] [-K keyfile] [ -N key_name ]
[-p port] [-v] [-w adminPW | ?] [-Z] [-?]
command -- [ibmslapd options]
```

Where *command* is {start|stop|restart|status|admstop}.

Description

The administration daemon control program, `ibmdirctl`, is used to start, stop, restart or query the status of the IBM Tivoli Directory Server. It can also be used to stop the administration daemon.

To display syntax help for `ibmdirctl`, type `ibmdirctl -?`.

Options

The options are:

► -D adminDN

   Use adminDN to bind to the LDAP directory. The adminDN is a string-represented DN (see LDAP Distinguished Names).

► -h hostname

   Specify an alternate host on which the ldap server and the admin daemon are running.

► -K keyfile

   Specifies the file to use for keys.

► -N key_name

   Specifies the private key name to use in keyfile.

► -p port

   Specify an alternate TCP port where the admin daemon is listening. The default LDAP port is 3538.

► -v

   Specifies to run in verbose mode.

► -w *adminPW* | ?

   Use `adminPW` as the password for authentication. Use the ? to generate a password prompt. Using this prompt prevents your password from being visible through the **ps** command. Refer to

► -?

   Displays the help screen

► command

   – start - Starts the server
   – stop - Stops the server
   – restart - Stops then starts the server
   – status - Queries the status the server
   – admstop - Stops the IBM Tivoli Directory Server administration daemon

   > **Note:** The stop command may be issued directly to the LDAP server.
   >
   > If the **admstop** command is issued successfully, the IBM Tivoli Directory Server Administration Daemon must be restarted manually.

► -- ibmslapd options

   The `ibmslapd` options are any options the `ibmslapd` process takes at startup time, typically:

   – -a | -A - Starts the server in configuration only mode
   – -n | -N - Does not start the server, if the server is unable to start with the database backends (no configuration only mode)

   > **Note:** If `ibmslapd` options are requested, they must be preceded by the `--`. The `ibmslapd` options are ignored if the `stop` command is issued.

   > **Note:** The -n and -N options prevent the server from starting if the server is unable to start with the database backends (not in configuration only mode).

To start the server in configuration-only mode issue the command:

```
ibmdirctl -h mymachine -D myDN -w mypassword -p 3538 start -- -a
```

To stop the server issue the command:

```
ibmdirctl -h mymachine -D myDN -w mypassword -p 3538 stop
```

To stop and start the LDAP Server with out showing the password:

```
C:\>ibmdirctl -D cn=root -w ? stop
Enter password ==> <password is not shown when typed>
Stop operation succeeded
C:\>ibmdirctl -D cn=root -w ? start
```

```
Enter password ==> <password is not shown when typed>
Start operation succeeded
```

9.14 Manual installation of IBM WAS - Express

If you use the InstallShield GUI to install the Web Administration Tool, you can select the embedded version of IBM WebSphere Application Server - Express for installation. In this case, configuration is also done automatically. If you use native installation methods, you can install and configure the embedded version of IBM WebSphere Application Server - Express manually. If you already have the embedded version of IBM WebSphere Application Server - Express V5.0.2 installed, you must configure manually before you can use the Web Administration Tool.

9.14.1 Manually installing the Web Administration Tool

To manually install the embedded version of WebSphere Application Server - Express, use the following procedure:

1. After you download and unzip (or untar) the IBM Tivoli Directory Server zip or tar file, change directories to the directory where you expanded the file.

2. Type the following command at a command prompt:

    On Windows platforms:

    ```
    install.bat -installRoot embWASE_installpath -hostName localhost
    ```

    On UNIX platforms:

    ```
    install.sh -installRoot embWASE_installpath -hostName localhost
    ```

    Where *embWASE_installpath* is the directory where you are installing the embedded version of IBM WebSphere Application Server - Express. By convention, this directory is the appsrv subdirectory of the directory where IBM Tivoli Directory Server is installed, but you can use any directory.

After installing the Web Administration Tool, copy the Web Administration Tool to the embedded version of IBM WebSphere Application Server - Express directory by using the following commands:

```
mkdir embWASE_installpath/installableApps/
cp installpath/idstools/IDSWebApp.war embWASE_installpath/installableApps/
```

Where:

► *embWASE_installpath* is the directory where you are installing the embedded version of WebSphere Application Server - Express.

► *installpath* is the directory where IBM Tivoli Directory Server is installed.

Install the Web Administration Tool into the embedded version of IBM WebSphere Application Server - Express by using the following command:

► On Windows systems:

> **Note:** Type the command on one line.

```
"embWASE_installpath\bin\wsadmin.bat" -conntype NONE -c "$AdminApp \
install {embWASE_installpath\installableApps\IDSWebApp.war} \
{-configroot \"embWASE_installpath\config\" \
-node DefaultNode -usedefaultbindings -nodeployejb -appname \
IDSWebApp.war -contextroot \"IDSWebApp\"}"
```

► On UNIX systems:

> **Note:** Type the command on one line.

```
embWASE_installpath/bin/wsadmin.sh -conntype NONE -c "\$AdminApp \
install {embWASE_installpath/installableApps/IDSWebApp.war} \
{-configroot \"embWASE_installpath/config\" \
-node DefaultNode -usedefaultbindings -nodeployejb -appname \
IDSWebApp.war -contextroot \"IDSWebApp\"}"
```

> **Note:** If you install the Web Administration Tool and the embedded version of WebSphere Application Server - Express through the InstallShield GUI, these commands are run automatically.

9.14.2 Manually uninstalling the Web Administration Tool

To manually uninstall Web Administration Tool from the embedded version of IBM WebSphere Application Server - Express, use the following procedure:

1. Be sure that the application server is started.

2. Type the following at a command prompt to uninstall the Web Administration Tool:

   – On Windows platforms:

> **Note:** Type the command on one line.

```
embWASE_installpath\bin\wsadmin.bat -conntype NONE -c "$AdminApp \
uninstall IDSWebApp.war"
```

– On UNIX platforms:

> **Note:** Type the command on one line.

```
embWASE_installpath/bin/wsadmin.sh -conntype NONE -c "\$AdminApp
uninstall IDSWebApp.war"
```

Where *embWASE_installpath* is the path where you installed the embedded version of WebSphere Application Server - Express.

9.14.3 Default ports used by IBM WAS - Express

The embedded version of WebSphere Application Server - Express uses four default port settings:

```
Http Transport (port 1): 9080
Http Transport (port 2): 9443
Bootstrap/rmi port: 2809
Soap connector port: 8880
```

If a conflict exists with another application using one or more of these default ports, you can use a text editor to change from the default ports to unused ports.

Http Transport port 1

Find the line containing the port number 9080 in the following files and replace the 9080 with the port number that you want:

```
$WASHOME\appsrv\config\cells\DefaultNode\nodes\DefaultNode\servers\server1\
server.xml
$WASHOME\appsrv\config\cells\DefaultNode\virtualhosts.xml
```

Where $WASHOME is the directory where the embedded version of WebSphere Application Server - Express is installed.

Http Transport port 2

Find the line containing the port number 9443 in the following files and replace the 9443 with the port number that you want:

```
$WASHOME\config\cells\DefaultNode\nodes\DefaultNode\servers\server1\server.
xml
$WASHOME\config\cells\DefaultNode\virtualhosts.xml
```

Where $WASHOME is the directory where the embedded version of WebSphere Application Server - Express is installed.

Bootstrap/rmi port

Find the line containing the port number 2809 in the following file and replace the 2809 with the port number that you want:

```
$WASHOME\config\cells\DefaultNode\nodes\DefaultNode\serverindex.html
```

Where WASHOME is the directory where the embedded version of WebSphere Application Server - Express is installed.

Soap connector port

Find the line containing the port number 8880 in the following file and replace the 8880 with the port number that you want:

```
$WASHOME\config\cells\DefaultNode\nodes\DefaultNode\serverindex.html
```

Where WASHOME is the directory where the embedded version of WebSphere Application Server - Express is installed.

HTTP and HTTPS Ports

The embedded version of WebSphere Application Server - Express, V5.0.2 comes with HTTPS set up by default on port 9443. To use HTTPS, you must change your login URL to the following:

```
https://<hostname>:9443/IDSWebApp/IDSjsp/Login.jsp
```

For non-HTTPS connections, continue to use the URL:

```
http://<hostname>:9080/IDSWebApp/IDSjsp/Login.jsp
```

Additionally, if you want to change the application server's SSL certificate, you can create new key and trust store database files for the embedded version of WebSphere Application Server - Express to use. By default, the key and trust store database files are separate and are located in the <WASHOME>/etc directory. These files are named DummyServerKeyFile.jks and DummyServerTrustFile.jks respectively.

After you have created your new Java keystore files, you can change the key and trust store database files that WAS uses by modifying the <WASHOME>/config/cells/DefaultNode/security.xml file to use your new file names, passwords, and file formats. In Example 9-9, refer to the highlighted lines that indicate what gets modified in the security.xml file.

Example 9-9   security.xml file

```
<repertoire xmi:id="SSLConfig_1" alias="DefaultSSLSettings">
        <setting xmi:id="DefaultSSLSettings"
keyFileName="${USER_INSTALL_ROOT}/etc/DummyServerKeyFile.jks"
keyFilePassword="WebAS" keyFileFormat="JKS"
```

```
trustFileName="${USER_INSTALL_ROOT}/etc/DummyServerTrustFile.jks"
trustFilePassword="WebAS" trustFileFormat="JKS"
clientAuthentication="false" securityLevel="HIGH"
enableCryptoHardwareSupport="false">
<cryptoHardware xmi:id="CryptoHardwareToken_1" tokenType=""
libraryFile="" password=""/>
<properties xmi:id="Property_4" name="com.ibm.ssl.protocol" value="SSLv3"/>
<properties xmi:id="Property_5" name="com.ibm.ssl.contextProvider"
value="IBMJSSE"/>
</setting>
</repertoire>
```

9.15 Installing in WebSphere Version 5.0 or later

If you use WebSphere, you must install the Web Administration Tool into Web-Sphere. Use the following instructions as a guide:

1. Install WebSphere, using the installation information provided with it.

2. Install the Web Administration Tool using either the InstallShield GUI or the installation utility for your operating system. The file containing the Web Administration Tool is named IDSWebApp.war, and it is in the idstools subdirectory of the installation directory that was specified during installation.

3. Install the Web Administration Tool application into WebSphere, using the information provided with WebSphere. For example, if you use the Administrative Console, on the Install New Application window, set the Local path to installdirectory/idstools/IDSWebApp.war, and the Context root to /IDSWebApp. installdirectory is the directory you specified when installing the Web Administration Tool.

4. Start the Web Administration Tool (for example, through the Administrative Console).

5. From a Web browser, type the following address:

    ```
    http://localhost:9080/IDSWebApp/IDSjsp/Login.jsp
    ```

    The IBM Tivoli Directory Server Web Administration login page window is displayed.

Note: This address works only if you are running the browser on the computer on which the Web Administration Tool is installed. If the Web Administration Tool is installed on a different computer, replace `localhost` with the hostname or IP address of the computer where the Web Administration Tool is installed.

One problem found was if you are running the browser on the computer on which the Web Administration Tool is installed and you are using an IP address or hostname as part of the URL used to access the Web Administration Tool you might get errors trying to connect. To fix this use localhost:9080 instead of the IP address or hostname and you will not have any problems.

10

Client tools

Normally we have different ways of performing certain activities. We can either use the command-line utilities or we can use a GUI for similar activities. A GUI or a graphical user interface is a handy tool in cases like:

► You have taken up a new product for study/use.

► The command-line utilities fail to give a proper understanding of the system topology.

The GUI provides the user a graphical feel of a product. For example, if a person wants to see the topology of a set of interconnected systems, he can have a much better view of the same through the GUI, rather than on the command line. However, there are situations where the command-line utilities seem to dominate over the GUI. Some of the disadvantages of the GUI can be listed as:

► Very little chances of automation

► Slow response

These drawbacks are overcome using the command line utilities.The command line utilities can be judiciously incorporated in scripts to have the desired tasks/tests automated. The responses from the command line clients are much faster as regards their GUI counterparts.

Let us talk in terms of the IBM Tivoli Directory Server. Suppose you want to continuously monitor the directory server, for the number of operations completed at a given instant of time. It will not be a good idea to manually refresh

the Web Administration page every 10–15 seconds, to see what the number of completed operations are at different instants of time. However, it would be a good idea to put the monitor search (`ldapsearch -D<admin DN> -w <admin PW> -s base -b cn=monitor objectclass=* | grep -i operations`) in a shell script, set a delay of 10–15 seconds, and allow it to run for the duration you want. No more user intervention is required and the results can be stored in a file, which can be analyzed at will. There are a lot more advantages of using the command-line utilities. We will be seeing these advantages in this chapter.

To begin with let us see what clients are shipped with the directory server and what can be done using them.

The client tools for the IBM Tivoli Directory Server come in two flavors. You can have the GUI as well as the command line utilities.

As far as the GUI is considered, the ITDS 5.2 Web Administration tool, which is shipped along with the product, acts as the graphical client for the directory server. However, that will not be explained here in its entirety. The relevant chapters will keep referring to the ways in which a particular activity can be done using the Web Administration tool. This chapter will mainly focus on the command line client utilities.

The LDAP clients that we will be seeing in this chapter are:

► `ldapchangepwd`
► `ldapdelete`
► `ldapexop`
► `ldapmodify` and `ldapadd`
► `ldapmodrdn`
► `ldapsearch`

10.1 The ldapchangepwd command

ldapchangepwd is the command line tool for modifying a user's password. Here is the synopsis of the **ldapchangepwd** command.

10.1.1 Synopsis

```
ldapchangepwd -D binddn -w passwd | ? -n newpassword | ? [-C charset] [-d
debuglevel][-G realm][-h ldaphost] [-K keyfile] [-m mechanism] [-M] [-N
certificatename] [-O maxhops] [-p ldapport] [-P keyfilepw] [-R] [-U username]
[-v] [-V version] [-y proxydn] [-Y] [-Z] [-?]
```

10.1.2 Options

The options are:

▶ -C *charset*

Specifies that the DNs supplied as input to the `ldapchangepwd` utility are represented in a local character set, as specified by charset. Use -C *charset* to override the default, where strings must be supplied in UTF-8. You may refer the ITDS 5.2 Administration Guide to get to know the character sets that we support.

You can download the administration guide from:

`http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html`

> **Note:** The supported values for charset are the same values supported for the charset tag that is optionally defined in Version 1 LDIF files.

▶ -d *debuglevel*

Set the LDAP debugging level to *debuglevel*. You may refer Chapter 18, "Debugging IBM Tivoli Directory Server related issues" on page 589, for further information on what debugging levels can be set for the clients. The level is the same as applicable to `ibmslapd` (the directory server process) while running it in debug mode.

▶ -D *binddn*

Use *binddn* to bind to the LDAP directory. *binddn* is a string-represented DN. When used with -m DIGEST-MD5, it specifies the authorization ID. It can be either a DN or an authorized string that starts with `u:` or `dn:`.

▶ -G *realm*

Specify the name of the *realm*. When used with the -m DIGEST-MD5, the value is passed to the server during the bind.

► -h *ldaphost*

Specify an alternate host on which the LDAP server is running. This option is useful in the event that your client is installed on a different system than directory server.

► -K *keyfile*

Specify the name of the SSL or TLS key database file with default extension of kdb. If the key database file is not in the current directory, specify the fully qualified key database filename. If a key database filename is not specified, this utility will first look for the presence of the SSL_KEYRING environment variable with an associated filename. If the SSL_KEYRING environment variable is not defined, the default keyring file will be used, if present.

A default keyring file, ldapkey.kdb, and the associated password stash file, ldapkey.sth, are installed in the /lib directory under LDAPHOME, where LDAPHOME is the directory where the directory server was installed. LDAPHOME varies by operating system platform:

– AIX operating systems - /usr/ldap
  • HP-UX operating systems - /usr/IBMldap
– Linux operating systems - /usr/ldap
– Solaris operating systems - /opt/IBMldapc
– Windows operating systems - C:\Program Files\IBM\LDAP

> **Note:** This is the default install location. The actual LDAPHOME is determined during installation. Currently it is possible to specify a different installation path only for Solaris and Windows. The other platforms are mandatorily installed at the default location.

See *IBM Directory C-Client SDK Programming Reference* for more information about default key database files, and default Certificate Authorities. This document can be downloaded from:

http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html

If a keyring database file cannot be located, a "hard-coded" set of default trusted certificate authority roots is used. The key database file typically contains one or more certificates of certificate authorities (CAs) that are trusted by the client. These types of X.509 certificates are also known as trusted roots. For more information on managing an SSL or TLS key database, refer to Chapter 15, "Securing the directory" on page 431.

This parameter effectively enables the -Z switch.

- ► -m *mechanism*

  Use *mechanism* to specify the SASL mechanism to be used to bind to the server. The ldap_sasl_bind_s() API will be used. The -m parameter is ignored if -V 2 is set. If -m is not specified, simple authentication is used.

- ► -M

  Manage referral objects as regular entries.

- ► -n *newpassword | ?*

  Specifies the new password. Use the ? to generate a password prompt. Using this prompt prevents your password from being visible through the `ps` command.

- ► -N *certificatename*

  Specify the label associated with the client certificate in the key database file. If the LDAP server is configured to perform server authentication only, a client certificate is not required. If the LDAP server is configured to perform client and server authentication, a client certificate might be required. *certificatename* is not required if a certificate/private key pair has been designated as the default. Similarly, *certificatename* is not required if there is a single certificate/private key pair in the designated key database file. This parameter is ignored if neither -Z nor -K is specified.

- ► -O *maxhops*

  Specify *maxhops* to set the maximum number of hops that the client library takes when chasing referrals. The default hopcount is 10.

- ► -p *ldapport*

  Specify an alternate TCP port where the LDAP server is listening. The default LDAP port is 389. If -p is not specified and -Z is specified, the default LDAP SSL port 636 is used.

- ► -P *keyfilepw-*

  Specify the key database password. This password is required to access the encrypted information in the key database file, which may include one or more private keys. If a password stash file is associated with the key database file, the password is obtained from the password stash file, and the -P parameter is not required. This parameter is ignored if neither -Z nor -K is specified.

- ► -R

  Specifies that referrals are not to be automatically followed.

- ▶ -U *username*

  Specifies the *username*. This is required with -m DIGEST-MD5 and ignored when any other mechanism is used. The value *username* depends on what attribute the server is configured to use. It might be a uid or any other value that is used to locate the entry.

- ▶ -v

  Use verbose mode, with many diagnostics written to standard output.

- ▶ -V *version*

  Specifies the LDAP *version* to be used by `ldapdchangepwd` when it binds to the LDAP server. By default, an LDAP V3 connection is established. To explicitly select LDAP V3, specify -V 3. Specify -V 2 to run as an LDAP V2 application. An application, like `ldapdchangepwd`, selects LDAP V3 as the preferred protocol by using `ldap_init` instead of `ldap_open`.

- ▶ -w *passwd | ?*

  Use passwd as the password for authentication. Use the *?* to generate a password prompt. Using this prompt prevents your password from being visible through the ps command.

- ▶ -y *proxydn*

  Specifies the DN to be used for proxied authorization. You can refer the section on `ldapsearch` for an example of using the -y option.

- ▶ -Y

  Use a secure TLS connection to communicate with the LDAP server. The -Y option is only supported when IBM's GSKit, is installed.

- ▶ -Z

  Use a secure SSL connection to communicate with the LDAP server. The -Z option is only supported when the SSL component entry, as provided by IBM's GSKit, is installed.

- ▶ -?

  Displays the syntax help for `ldapchangepwd`.

10.1.3 Examples

The following examples illustrate the options that we have just discussed.

Example 1
The following command:

```
C:\>ldapchangepwd -D cn=user1,o=ibm,c=us -w user -n user1
```

Changes the password of the entry with commonName "user1" from user to user1.

Example 2

Here is an example for using a different *charset* than the default. First, let us verify the codepage of the current database. You can do so with the following instructions on Windows.

On Windows, you need to work in the DB2 shell, invoked by running the **db2cmd** command. In that shell, here are the commands to use:

```
C:\>set DB2INSTANCE=ldapdb2
C:\>db2start
C:\>db2 connect to ldapdb2
   Database Connection Information
   Database server        = DB2/NT 8.1.2
   SQL authorization ID   = ADMINIST...
   Local database alias   = LDAPDB2
C:\>db2 get db cfg for ldapdb2 | grep -i code
   Database code page                              = 1208
   Database code set                               = UTF-8
   Database country/region code                    = 1
```

The output from these commands show that the current database is UTF-8.

> **Note:** We have used a UNIX utility here called `grep`, which is not available on Windows by default. You can take the entire output in a file and search for the relevant lines.

The next command:

```
C:\>ldapchangepwd -D cn=user1,o=ibm,c=us -w user1 -C ISO-8859-1 -n user2
changing password for entry cn=user1,o=ibm,c=us
```

Changes the password of the entry with the commonName "user1" from user1 to user2. Note that `ldapchangepwd` tells the server that the dn that is passed to it was specified in the ISO-8859-1 character set.

Example 3

Let us run the **ldapchangepwd** command with the minimum *debuglevel* of 1, just to see the kind of debug output it shows up. Here is what you get when you try to use invalid credentials to change the password:

```
C:\>ldapchangepwd -D cn=user1,o=ibm,c=us -w user2 -d 1 -n user3
      065:19:42:30 T2652 ldap_sasl_bind
      065:19:42:30 T2652 ldap_sasl_bind_direct
      065:19:42:30 T2652 put_ctrls_into_ber: ctrls=00304F88
```

```
065:19:42:30 T2652 put_ctrls_into_ber: return(rc=0)
065:19:42:30 T2652 send_initial_request
065:19:42:30 T2652 open_default_connection
065:19:42:30 T2652 new_connection: connect=1
065:19:42:30 T2652 open_ldap_connection
065:19:42:30 T2652 connect_to_host: rainbow:389
065:19:42:30 T2652 sd 716 connected to: 127.0.0.1
065:19:42:30 T2652 new_connection: successful - return(lc=00306520)
065:19:42:30 T2652 send_server_request: msgid=1, bind=NONE
065:19:42:30 T2652 use_connection: lc=00306520, new refcount=2
065:19:42:30 T2652 flush_request: msgid=1
065:19:42:30 T2652 do_ldap_select
065:19:42:30 T2652 ldap_result
065:19:42:30 T2652 wait4msg (infinite timeout)
065:19:42:30 T2652 do_ldap_select
065:19:42:30 T2652 read1msg
065:19:42:30 T2652 got result msgid 1, original id 1
065:19:42:30 T2652 free_request (origid 1, msgid 1)
065:19:42:30 T2652 free_connection: lc=00306520, force=0, unbind=1
065:19:42:30 T2652 free_connection: lc=00306520, not freed, refcnt 1
065:19:42:30 T2652 get_ctrls_from_ber: ctrls_p=0012FE78
065:19:42:30 T2652 get_ctrls_from_ber: Control OID =
1.3.6.1.4.1.42.2.27.8.5.1, critical = No, value follows
065:19:42:30 T2652 get_ctrls_from_ber: control value is NULL.
065:19:42:30 T2652 get_ctrls_from_ber: return(0), ctrls=00306060, 1
controls returned
065:19:42:30 T2652 ldap_msgfree
065:19:42:30 T2652 ldap_controls_free: ctrls=00304F88
065:19:42:30 T2652 ldap_control_free: ctrl=00304F48
065:19:42:30 T2652 ldap_controls_free: ctrls=00000000
065:19:42:30 T2652 ldap_err2string
ldap_simple_bind:   Invalid credentials
```

Example 4

Here is an example where you use the -*h hostname* argument to indicate the host
where the password change is expected:

```
C:\>ldapchangepwd -h localhost -D cn=user1,o=ibm,c=us -w user5 -n user6
changing password for entry cn=user1,o=ibm,c=us
```

Since the default host is localhost, this command is the same as the one shown
in "Example 1" on page 242.

Example 5

This example shows the way password changes are done over SSL:

```
C:\>ldapchangepwd -D cn=user1,o=ibm,c=us -K F:\KEYS\clientCMS.kdb -P client
-Z -w user6 -n user7
```

```
changing password for entry cn=user1,o=ibm,c=us
```

Here the path of the key file is passed using the -K option, the keyfile password is passed using the -P option and the SSL flag is turned on using the -Z option (this is optional in the example shown above, as -K is supposed to enable the -Z switch by default).

Example 6

This example shows the use of the -N option.

```
C:\>ldapchangepwd -D cn=user1,o=ibm,c=us -K
f:\Ramakrishna\KEYS\clientCMS.kdb -P client -Z -N client -w user7 -n user8
ldap_simple_bind:   Operations error
changing password for entry cn=user1,o=ibm,c=us
Can't contact LDAP server

C:\>ldapchangepwd -D cn=user1,o=ibm,c=us -K
f:\Ramakrishna\KEYS\clientCMS.kdb -P client -Z -N clientCMS -w user7 -n
user8
changing password for entry cn=user1,o=ibm,c=us
```

This example shows that you are not allowed to change the password in case you pass the wrong certificate name. In "Example 5" on page 244, the client was picking up the correct certificate to talk to the server, as that was the only one available and which acted as the default one.

Example 7

This example shows the use of the -p option:

```
C:\>ldapchangepwd -D cn=user1,o=ibm,c=us -p 389 -w user8 -n user9
changing password for entry cn=user1,o=ibm,c=us
```

As seen above the port, over which ldapchangepwd is talking to the server, is 389. This happens to be the default port. This is configurable, and in the cases where the default port was changed, the -p option is needed.

Example 8

This example shows the use of flags/options pertaining to referrals. We will get to know the details on using the -O option, here. You may refer the ldapsearch section for illustrations on the -M and -R options.

Assuming we have a set of referrals pointing to an entry, as follows:

cn=ref1,o=ibm,c=us -> o=ref,o=ibm,c=us -> cn=user1,o=ibm,c=us

ref1 is a referral to ref, which in turn is a link to cn=user1.

If the `ldapchangepwd` is run on the entry `cn=ref1,o=ibm,c=us` with number of hops =1:

```
C:\>ldapchangepwd -D cn=ref1,o=ibm,c=us -p 389 -w user9 -O 1 -n user1
ldap_simple_bind:   Referral limit exceeded
```

The example shows that `cn=ref1,o=ibm,c=us` could not reach the actual target `cn=user1,o=ibm,c=us` in the specified number of hops (1).

Now let us remove the restrictions on the referrals:

```
C:\>ldapchangepwd -D cn=ref1,o=ibm,c=us -p 389 -w user9 -n user1
changing password for entry cn=ref1,o=ibm,c=us
```

Now the password change takes place successfully.

Example 9

The next example shows the **ldapchangepwd** command driven in verbose (-v) mode:

```
C:\>ldapchangepwd -D cn=ref1,o=ibm,c=us -p 389 -v -w user1 -n user2
ldap_init(NULL, 389)
changing password for entry cn=ref1,o=ibm,c=us
delete userpassword:
        user1
add userpassword:
        user2
ldapchangepwd complete
```

The example shows a more detailed way as to how **ldapchangepwd** goes about changing the password of a specific user.

Example 10

This example shows the usage of the -V option:

```
C:\>ldapchangepwd -D cn=ref1,o=ibm,c=us -V 2 -w user1 -n user2
ldap_bind_s: Inappropriate authentication

C:\>ldapchangepwd -D cn=ref1,o=ibm,c=us -V 3 -w user2 -n user3
changing password for entry cn=ref1,o=ibm,c=us
```

As shown above the server refuses the client any service, saying `Inappropriate Authentication`, as it is expecting a version 3 call from **ldapchangepwd**.

Example 11

This example shows the usage of the *-w password | ?* and *-n newpassword | ?* options in place of the password to avoid entering them on the command line:

```
C:\>ldapchangepwd -D cn=ref1,o=ibm,c=us -w ? -n ? -v
```

```
Enter Old password ==>
Enter New password ==>
ldap_init(NULL, 389)
changing password for entry cn=ref1,o=ibm,c=us
delete userpassword:
        user3
add userpassword:
        user4
ldapchangepwd complete
```

The verbose mode is deliberately turned on here to show how the change in the password is taking place. In case the passwords are entered along with the command (without the ? option), the passwords remain in the history of the shell and it is possible for other users to go through the history and get the passwords. Also the **ps** command would be showing the password. To overcome such issues option of ? is used.

Example 12

The next example shows how the user's password may be changed over TLS:

```
C:\>ldapchangepwd -D cn=user1,o=ibm,c=us -Y -w user6 -n user7 -K
F:\Ramakrishna\KEYS\clientCMS.kdb -P client
changing password for entry cn=user1,o=ibm,c=us
```

The server should be capable of accepting TLS connections in this case.

The root DSE search can be used to verify this:

```
C:\>ldapsearch -s base objectclass=* | grep security
security=tls
```

You may refer Chapter 15, "Securing the directory" on page 431, for further information on TLS.

The *-G realm* option is effective only when you have set up SASL communications. That is, it goes hand-in-hand with the *-m mechanism* option. Same is the case with the *-U username* option. The *-U* options is ignored if *-m* option is not specified in the command line. More information on Realms can be had from the ITDS 5.2 Administration Guide. This document can be found at:

http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html

As far as the SASL mechanisms considered, you can have more information on the same by reading Chapter 15, "Securing the directory" on page 431.

10.1.4 SSL, TLS notes

To use the SSL or TLS - related functions associated with this utility, the SSL or TLS libraries and tools must be installed. The SSL or TLS libraries and tools are provided with IBM's Global Security Kit (GSKit), which includes security software developed by RSA Security Inc.

> **Note:** For information regarding the use of 128-bit and triple DES encryption by LDAP applications, including the LDAP sample programs, see "LDAP_SSL" in the *IBM Directory C-Client SDK Programming Reference*. This section describes the steps required to build the sample programs and your applications so that they can use SSL with the strongest encryption algorithms available. This document can be downloaded from:
>
> http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html

See the makefile associated with the sample programs for more information on linking an LDAP application so that it has access to 128-bit and triple-DES encryption algorithms.

The content of a client's key database file is managed with the `gsk7ikm` utility. For more information on this Java utility, please refer Chapter 15, "Securing the directory" on page 431. The `gsk7ikm` utility is used to define the set of trusted certification authorities (CAs) that are to be trusted by the client. By obtaining certificates from trusted CAs, storing them in the key database file, and marking them as "trusted", you can establish a trust relationship with LDAP servers that use "trusted" certificates issued by one of the trusted CAs. The `gsk7ikm` utility can also be used to obtain a client certificate, so that client and server authentication can be performed.

If the LDAP servers accessed by the client use server authentication only, it is sufficient to define one or more trusted root certificates in the key database file. With server authentication, the client can be assured that the target LDAP server has been issued a certificate by one of the trusted CAs. In addition, all LDAP transactions that flow over the SSL or TLS connection with the server are encrypted including the LDAP credentials that are supplied on the `ldap_bind` or `ldap_simple_bind_s`. For example, if the LDAP server is using a high-assurance VeriSign certificate, you should obtain a CA certificate from VeriSign, import it into your key database file, and mark it as trusted. If the LDAP server is using a self-signed server certificate, the administrator of the LDAP server can supply you with a copy of the server's certificate request file. Import the certificate request file into your key database file and mark it as trusted.

If the LDAP servers accessed by the client use client and server authentication, it is necessary to:

- ▶ Define one or more trusted root certificates in the key database file. This allows the client to be assured that the target LDAP server has been issued a certificate by one of the trusted CAs. In addition, all LDAP transactions that flow over the SSL or TLS connection with the server are encrypted, including the LDAP credentials that are supplied on the `ldap_bind` or `ldap_simple_bind_s`.

- ▶ Create a key pair using `gsk7ikm` and request a client certificate from a CA. After receiving the signed certificate from the CA, store the certificate in the client key database file.

10.1.5 Diagnostics

The exit status is 0 if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error.

10.2 The ldapdelete command

`ldapdelete` is the command-line tool for deleting a single or a group of users/entries. The entries to be deleted can be passed through the command line or through file redirection. Let us go further and see the detailed synopsis of the **ldapdelete** command.

10.2.1 Synopsis

```
ldapdelete [-c] [-C charset] [-d debuglevel][-D binddn] [-f file] [-G realm]
[-h ldaphost] [-i file] [-k] [-K keyfile] [-m mechanism] [-M] [-n] [-N
certificatename] [-O maxops] [-p ldapport] [-P keyfilepw] [-R] [-s][-U
username} [-v] [-V version] [-w passwd | ?] [-y proxydn][-Y] [-Z] [dn]...
```

10.2.2 Description

`ldapdelete` is a command-line interface to the `ldap_delete` library call.

`ldapdelete` opens a connection to an LDAP server, binds, and deletes one or more entries. If one or more Distinguished Name (DN) arguments are provided, entries with those DNs are deleted. Each DN is a string-represented DN. If no DN arguments are provided, a list of DNs is read from standard input, or from a file, if the `-i` flag is used.

To display syntax help for ldapdelete, type:

```
ldapdelete -?.
```

10.2.3 Options

`ldapdelete` has options/arguments like -C *charset*, -d *debuglevel*, -D *binddn*, -G *realm*, -h *ldaphost*, -K *keyfile*, -m *mechanism*, -M, -N *certificatename*, -O *maxhops*, -p *ldapport*, -P *keyfilepw*, -R, -U *username*, -v, -V, -w *passwd* / *?*, -y *proxydn*, -Y, -Z, which are the same as the **ldapchgpwd** command and have been explained in "Options" on page 239. Therefore they are not explained any further here. The command line arguments for `ldapdelete` that We will be seeing, in this section, are as follows:

► -c

  Continuous operation mode. Errors are reported, but `ldapdelete` continues with modifications. Otherwise the default action is to exit after reporting an error.

► -f *file*

  Read a series of lines from a file, performing one LDAP delete for each line in the file. Each line in the file should contain a single distinguished name.

► -i *file*

  Read a series of lines from a file, performing one LDAP delete for each line in the file. Each line in the file should contain a single distinguished name.

► -k

  Specifies to use server administration control.

► -n

  Show what would be done, but do not actually modify entries. Useful for debugging in conjunction with -v.

► -s

  Use this option to delete the subtree rooted at the specified entry.

► -dn

  Specifies one or more DN arguments. Each DN should be a string-represented DN.

10.2.4 Examples

Lets see a set of examples illustrating the options/arguments that we have just discussed above.

Example 1
The following command:

```
ldapdelete "cn=Delete Me, o=University of Life, c=US"
```

Attempts to delete the entry with commonName "Delete Me" directly below the "University of Life" organizational entry. It might be necessary to supply a binddn and passwd, for deletion to be allowed (see the -D and -w options).

Example 2

Here is an example of using the -c argument.

Assuming the fact that there exists an LDIF file test.ldif, with the contents:

```
cn=user10,o=ibm,c=us
cn=user4,o=ibm,c=us
```

And with the directory users as shown below:

```
C:\>ldapsearch -D cn=root -w secret -b o=ibm,c=us cn=user* dn
cn=user1,o=ibm,c=us
cn=user2,o=ibm,c=us
cn=user3,o=ibm,c=us
cn=user4,o=ibm,c=us
```

Now if the **ldapdelete** command is used without the -c option:

```
C:\>ldapdelete -D cn=root -w secret -f test.ldif
Deleting entry cn=user10,o=ibm,c=us
ldap_delete:  No such object

C:\>ldapdelete -D cn=root -w secret -c -f test.ldif
Deleting entry cn=user10,o=ibm,c=us
ldap_delete:  No such object
Deleting entry cn=user4,o=ibm,c=us
```

As seen and expected, -c did not break ldapdelete and it went ahead with the deletion of the rest of the entries specified in the file.

There are other ways of doing the same operation:

```
ldapdelete -D cn=root -w secret -c -i test.ldif
ldapdelete -D cn=root -w secret -c < test.ldif
```

Example 3

This example is an illustration of the Admin Control (-k). Assuming the fact that there exists a replication topology and attempts to delete some entries on the replica:

```
C:\>ldapdelete -D cn=root -w secret cn=user3,o=ibm,c=us
Deleting entry cn=user3,o=ibm,c=us
ldap_delete:  DSA is unwilling to perform
ldap_delete: additional info: Data not encrypted
Referral:
ldap://localhost:636
```

That is, with the normal set of options, we are not able to delete the entry. Now we can delete the same using the Admin Control as follows:

```
C:\>ldapdelete -D cn=root -w secret -k cn=user3,o=ibm,c=us
Deleting entry cn=user3,o=ibm,c=us
```

Example 4

This example shows the use of the -n option:

```
C:\>ldapdelete -D cn=root -w secret -n -dn cn=user1,o=ibm,c=us
!Deleting entry cn=user1,o=ibm,c=us

C:\>ldapsearch -D cn=root -w secret -b cn=user1,o=ibm,c=us objectclass=* dn
cn=user1,o=ibm,c=us
```

The example shows that the entry is not physically deleted.

Example 5

This example shows how the dn that is to be deleted can be passed in the same line as the ldapdelete command, without any input redirection from a file.

```
C:\>ldapdelete -D cn=root -w secret -k -dn cn=user3,o=ibm,c=us
Deleting entry cn=user3,o=ibm,c=us
```

Example 6

This example shows the use of the -s option to delete an entire subtree. Here is how that can be done.

Suppose we want to delete the subtree "cn=sub,o=ibm,c=us" with subentries as shown below:

```
C:\>ldapsearch -D cn=root -w secret -b cn=sub,o=ibm,c=us objectclass=* dn
cn=sub,o=ibm,c=us

cn=sub1,cn=sub,o=ibm,c=us
```

Now here is the command to delete the subtree in a single shot:

```
C:\>ldapdelete -D cn=root -w secret -s cn=sub,o=ibm,c=us
Deleting entry cn=sub,o=ibm,c=us

C:\>ldapsearch -D cn=root -w secret -b cn=sub,o=ibm,c=us objectclass=* dn
ldap_search: No such object
ldap_search: matched: O=IBM,C=US
```

Note: If no DN arguments are provided, the **ldapdelete** command waits to read a list of DNs from standard input. To break out of the wait, use Ctrl+C or Ctrl+D.

10.2.5 SSL, TLS notes

The SSL- or TLS-related functions associated with this utility are as like the ones described with `ldapchangepwd`.

10.2.6 Diagnostics

Exit status is 0 if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error.

10.3 The ldapexop command

`ldapexop` is the tool for performing the extended operations pertaining to the IBM Tivoli Directory Server.

10.3.1 Synopsis

```
ldapexop [-C charset] [-d debuglevel][-D binddn][-e] [-G realm] [-h ldaphost]
[-help][-K keyfile] [-m mechanism] [-N certificatename] [-p ldapport]
[-P keyfilepw] [-?] [-U username] [-v] [-w passwd | ?] [-Y] [-Z]

-op {cascrepl | clearlog | controlqueue | controlrepl | getAttributes |
getlogsize | getusertype | quiesce | readconfig | readlog | stopserver | unbind
| uniqueattr }
```

10.3.2 Description

The `ldapexop` utility is a command-line interface that provides the capability to bind to a directory and issue a single extended operation along with any data that makes up the extended operation value.

The `ldapexop` utility supports the standard host, port, SSL, TLS, and authentication options used by all of the LDAP client utilities. In addition, a set of options is defined to specify the operation to be performed, and the arguments for each extended operation

To display syntax help for `ldapexop`, type:

```
ldapexop -?
```

Or:

```
ldapexop -help
```

10.3.3 Options

The options for the **ldapexop** command are divided into two categories:

- ► General options that specify how to connect to the directory server. These options must be specified before operation specific options.

- ► Extended operation option that identifies the extended operation to be performed.

General options

These options specify the methods of connecting to the server and must be specified before the -op option.

ldapexop expects general options such as -C *charset*,-d *debuglevel*, -D *binddn*, -G *realm*, -h *ldaphost*, -*help*, -K *keyfile*, -m *mechanism*, -p *ldapport*,-P *keyfilepw*, -?, -U *username*, -v, -w *passwd | ?*, -Y, -Z which are the same as the **ldapchangepwd** command and have been explained in "Options" on page 239.

There is one general option that needs explanation here.

- ► -e

  This option displays the LDAP library version information and quits. Here is an example of the output:

  ```
  C:\>ldapexop -e
  SDK Version:          510
  Protocol Version:     300
  SDK Build Level:      Oct  1 2003
  ```

Extended operations option

The -op *extended-op* option identifies the extended operation to be performed. The extended operation can be one of the following values:

- ► cascrepl -action *<actionvalue>* -rc *<contextDN>* *[options]*

  This extended operation is for controlling the cascading replication. The requested action is applied to the specified server and also passed along to all replicas of the given subtree. If any of these are forwarding replicas, they forward the extended operation to their replicas.

  The operation cascades over the entire replication topology.

  -action *{quiesce | unquiesce | replnow | wait}*

  This is a required attribute that specifies the action to be performed.

  - – *quiesce*

    This operation indicates the server to take no further updates till the relevant subtree is unquiesced. While setting up the topology, it is desired

that the changes should not go through the servers participating in the topology, so that a data consistency is maintained across all the servers. Hence there is the option of quiescing the servers. The only way to make any updates to a quiesced server is through an Admin Control. The obvious reason for having the option of the Admin Control is that you need to write to the servers the replication related information and you need a channel to write to the servers even when they are quiesced. Hence the necessity and implementation of the Admin Control.

– unquiesce

Resume normal operation, client updates are accepted. Once a topology is completed the subtree (replication context) can be unquiesced. that is, It is ready to accept changes again.

– replnow

Replicate all queued changes to all replica servers as soon as possible, regardless of schedule. In other words this option triggers forceful replication.

– wait

Wait for all updates to be replicated to all replicas. In other words the topology will be in a sort of dormant stage or a sort of sleep mode, till the entire topology has come to a balanced or synchronized state. That is all the updates in the queues of the relevant Masters/Forwarders have gone in place.

– -rc *contextDn*

This is a required attribute that specifies the root of the subtree. rc stands for replication context. In case you have set up replication, you can edit the relevant subtree to find that the object class `ibm-replicationContext` is added to the subtree, say, for example, `o=ibm,c=us`, to make it eligible for replication. The term rc is picked up from this (r)eplication (c)ontext.

options

– -timeout *secs*

This is an optional attribute that if present, specifies the timeout period in seconds. If not present, or 0, the operation waits indefinitely. For better performance of your replication topology it is advisable to set a timeout period. Some of the servers in the topology may be down. Consequently the updates to these down servers may not be sent till the servers are up. Hence there is no point in waiting indefinitely for the changes to pass to all the servers. Keeping a timeout would mean that you are allocating the necessary resources for the necessary amount of time and not more. If there are any anomalies in the topology at a given instant of time, they can be detected using the other options of the **ldapexop** command.

For example:

```
ldapexop -op cascrepl -action -quiesce -rc "o=acme,c=us" -timeout 60
```

This command is meant to quiesce the subtree o=acme,c=us that is, prevent it from taking any further updates, other than from the administration control. The operation is supposed to quit if it does not complete in 60 seconds.

▶ clearlog -log *<logname>*

This extended operation is used to clear the log files from the command line. The log files which can be cleared by the **ldapexop** command are listed below, as an argument to the -log option to **ldapexop**.

`-log {audit | bulkload | cli | slapd | ibmdiradm | adminDaemon | debug}`

This is a required attribute that specifies which log file to be cleared. The parameters to the -log, as shown above, are mostly self-explanatory as to what it'll clear. The only log that needs a mention is the debug log. When you use ldapexop to clear the debug log then the file pointed to by LDAP_DEBUG_FILE is cleared. This environment variable is supposed to store the file name so that when you are collecting the server debug trace the same can be redirected to it. For example:

`ldapexop -op clearlog -log debug`

▶ controlqueue -skip *<skipvalue>* -ra *<agreementDN>*

This extended operation, as its name indicates, is used for controlling the replication queue, as identified by the replication agreement.

– -skip *{all | change-id}*:

This is a required attribute.

• *all*

This option indicates to skip all pending changes for this agreement.

• *change-id*

This option identifies the single change to be skipped. If the server is not currently replicating this change, the request fails. In other words, if there are 100 entries in the replication queue, you are allowed to skip just the 100th. entry in the queue. There is no direct option, whereby you can skip any nth. entry in the replication queue. The entry to be skipped always has to be the front of the queue.

– -ra *agreementDN*

This is a required attribute that specifies the DN of the (r)eplication (a)greement. The objectclass pertaining to the replication agreement is ibm-replicationAgreement. ra is derived from this objectclass.

For example:

```
ldapexop -op controlqueue -skip all -ra "cn=server3,
ibm-replicaSubentry=master1-id,ibm-replicaGroup=default,o=acme,c=us"

ldapexop -op controlqueue -skip 2185 -ra
"cn=server3,ibm-replicaSubentry=master1-id,ibm-replicaGroup=default,
o=acme,c=us"
```

► controlrepl -action *<actionvalue>* {-rc *<contextDN>* | -ra *<agreementDN>*}

This extended operation is useful for controlling the replication activities associated with a specific subtree or associated with a specific recipient of replication.

– -action *{suspend | resume | replnow}*

This is a required attribute that specifies the action to be performed. This option is used either to suspend, resume or forcefully replicate changes over a specific queue as identified by the other options passed.

– -rc *contextDn* | -ra *agreementDn*

The -rc contextDn is the DN of the replication context. The action is performed for all agreements for this context. The -ra agreementDn is the DN of the replication agreement. The action is performed for the specified replication agreement. that is, to say if the -rc option is specified then that will affect all the queues associated with this subtree. And if -ra is specified then only that queue which corresponds to this agreementDN, will be affected.

For example:

```
ldapexop -op controlrepl -action suspend -ra "cn=server3,
ibm-replicaSubentry=master1-id,ibm-replicaGroup=default,o=acme,c=us"
```

This is an example to suspend the replication activities associated with the queue, where the supplier is the one pointed to by master1-id and the recipient is the one pointed to be server3.

► getattributes -attrType *<type>* -matches bool *<value>*

This is an extended operation whereby we can fetch the attributes of a specific type as understood by the rest of the options that go along. Let us see the detailed specifics on the same:

-attrType {*operational* | *language_tag* | *attribute_cache* | *unique* | *configuration*}

This is a required parameter for the getattributes extended operation. It specifies type of attribute being requested.

– *operational*

These are the set of attributes tracking the operations of the directory server. The clients are not supposed to play around with these attributes, as doing so may force the server to give incorrect information. The examples of the operational attributes are the attributes pertaining to ACLs, the attributes storing the timestamps of different events, some attributes of password policy, etc. For more information refer to Chapter 11, "Schema management" on page 287.

– *language_tags*

The term, language tags, defines a mechanism that enables the directory to associate natural language codes with values held in a directory and enables clients to query the directory for values that meet certain natural language requirements.

Here is an example:

```
ldapsearch -b "o=ibm,c=us" (objectclass=organization)
description;lang-en
```

The server returns values of an attribute `description;lang-en`, but does not return values of an attribute `description` or `description;lang-fr`.

If a request is made specifying an attribute without providing a language code, then all attribute values regardless of their language code are returned. Further information on this, refer to ITDS v5.2 Administration Guide, which can be found at:

http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2. html

– *attribute_cache*

In ITDS 5.2, there is a new concept of an attribute cache. This is designed for enhanced performance of the directory server. The attribute cache will store information pertaining to attributes. There is a setting by means of which you can select/deselect the set of attributes that can be cached. Like if you wish that the uid attribute should be cached, whenever there is a search on uid, just make the necessary settings and information pertaining to uid would be cached. Hence next time you need to get a specific set of results the attribute cache will also be screened, thus enhancing performance.

– *unique*

Each attribute in the LDAP schema maps to a single table. By adding any attribute to the list of unique attributes, the relevant column in the table corresponding to the attribute is made unique. Consequently if you make postaladdress as unique, it will not be possible for more than one object, in this directory server, to have the same postaladdress.

– *configuration*

These are basically attributes pertaining to the configuration of the server.

► -matches bool *{true | false}*

Specifies whether the list of attributes returned matches the attribute type specified by the -attrType option.

For example:

```
ldapexop -op getattributes -attrType unique -matches bool true
```

Returns a list of all attributes that have been designated as unique attributes.

```
ldapexop -op getattributes -attrType unique -matches bool false
```

Returns a list of all attributes that have been not been designated as unique attributes.

► getlogsize -log *<logname>*

This extended operation is used to request log file size, precisely in terms of the number of lines in the log file. This is very much like the 'wc -l' command on UNIX. However, ldapexop is more sophisticated way of doing such things though as you need not specify the name of the log file, just the type of the log is sufficient.

-log *{audit | bulkload | cli | slapd | ibmdiradm | adminDaemon | debug}*

This is a required attribute that specifies the log file to be queried The size of the log file, in lines, is written to standard output.

For example:

```
ldapexop -op getlogsize -log slapd
2000 lines
```

► getusertype

This extended operation is used to get to know the profile/privileges of a given user. This extended operation returns the user type based on the bound DN. For example:

```
ldapexop - D <AdminDN> -w <Adminpw> -op getusertype
```

Returns:

```
User : root_administrator
Role(s) : server_config_administrator directory_administrator
```

The description on the "User type and user roles" for extended operations is coming up shortly in the same section.

▶ quiesce -rc *<contextDN>* [options]

quiesce or unquiesce subtree extended operation. This extended operation is used to quiesce or unquiesce the servers associated with a specific subtree. The subtree is indicated by the -rc parameter.

-rc *contextDN:* This is a required attribute that specifies the DN of the replication context (subtree) to be quiesced or unquiesced.

The option is -end. This is an optional attribute that if present, specifies to end the quiesce state, or in other words unquiesce the subtree. If not specified the default is to quiesce the subtree. For example:

```
ldapexop -op quiesce -rc "o=acme,c=us"
ldapexop -op quiesce -end -rc "o=ibm,c=us"
```

▶ readconfig -scope *<scopevalue>*

This extended operation helps the server to dynamically read updates to the configuration file. Let us go into the specifics of the same:

```
-scope { entire | single <entry DN> <attribute> | entry <entry DN> |
subtree <entry DN> }
```

This is a required attribute. This specifies the scope of the configuration file that needs to be re-read by the server, which is currently up.

– entire

This option specifies to reread the entire configuration file.

– single *<entry DN><attribute>*

This option specifies to read the single entry and attribute as specified.

– entry *<entry DN>*

This option specifies that the server is supposed to read the specified entry.

– subtree *<entry DN>*

This option specifies that the server is supposed to read the entry and the entire subtree under it.

For example:

```
ldapexop -op readconfig -scope entire
ldapexop -op readconfig -scope single "cn=configuration"
ibm-slapdAdminPW
```

By means of the above example you are asking the server to dynamically take note of the new admin password.

There is a set of attributes which can be dynamically re-read by the directory server. If you change any other attributes and even if you ask the server to re-read the entire configuration file, changes will not take effect dynamically.

The following is a list of attributes that can be changed dynamically. You do not have to restart the server for these changes to take effect:

- cn=Configuration

  - ibm-slapdadmindn
  - ibm-slapdadminpw
  - ibm-slapderrorlog
  - ibm-slapdpwencryption
  - ibm-slapdsizelimit
  - ibm-slapdsysloglevel
  - ibm-slapdtimelimit

- cn=Front End, cn=Configuration

  - ibm-slapdaclcache
  - ibm-slapdaclcachesize
  - ibm-slapdentrycachesize
  - ibm-slapdfiltercachebypasslimit
  - ibm-slapdfiltercachesize
  - ibm-slapdidletimeout

- cn=Event Notification, cn=Configuration

  - ibm-slapdmaxeventsperconnection
  - ibm-slapdmaxeventstotal

- cn=Transaction, cn=Configuration

  - ibm-slapdmaxnumoftransactions
  - ibm-slapdmaxoppertransaction
  - ibm-slapdmaxtimelimitoftransactions

- cn=ConfigDB, cn=Config Backends, cn=IBM Directory, cn=Schemas, cn=Configuration

  - ibm-slapdreadonly

- cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas, cn=Configuration

  - ibm-slapdbulkloaderrors
  - ibm-slapdclierrors
  - ibm-slapdpagedresallownonadmin
  - ibm-slapdpagedreslmt
  - ibm-slapdpagesizelmt
  - ibm-slapdreadonly
  - ibm-slapdsortkeylimit
  - ibm-slapdsortsrchallownonadmin
  - ibm-slapdsuffix

▶ readlog -log *<logname>* -lines *<value>*

This extended operation is used to read log files. This extended operation is used to read a set of lines from the relevant log files or read the entire file. Let us go into the further specifics pertaining to this option.

```
-log audit | bulkload | cli | slapd | ibmdiradm | debug
```

This is a required attribute that specifies the log file to be queried.

```
-lines {<first><last> | all}
```

This is a required attribute that specifies either the number of the first and the last lines to be read from the file or it specifies that all lines are to be read. Lines are numbered starting at 0. The specified lines are written to standard output. For example:

```
ldapexop -op readlog -log audit -lines 10 20
ldapexop -op readlog -log slapd -lines all
```

▶ stopserver

This extended operation helps in stopping the IBM Tivoli Directory Server. For example:

```
ldapexop -op stopserver
```

▶ unbind {-dn *<specificDN>* | -ip *<sourceIP>* | -dn *<specificDN>* -ip *<sourceIP>* | all}

This extended operation is useful for disconnecting connections based on DN, IP, DN/IP or all connections, as needed. All the connections without any operations and all connections with operations on the work queue are ended immediately. If a worker is currently working on a connection, it is ended as soon as the worker completes that one operation.

– -dn <specificDN>

By specifying just the dn, ldapexop will end a connection by DN only. This request results in the purging of all the connections bound on the specified DN.

– -ip *<sourceIP>*

By specifying just the IP, ldapexop will end a connection by IP only. This request results in the purging of all the connections from the specified IP source.

– -dn *<specificDN>* **-ip** *<sourceIP>*

By specifying both the dn and the ip, ldapexop will end a connection determined by a DN/IP pair. This request results in the purging of all the connections bound on the specified DN and from the specified IP source.

- -all

  Issues a request to end all the connections. This request results in the purging of all the connections except the connection from where this request originated. This attribute cannot be used with the -dn and/or -ip. parameters.

  The unbind option is mostly useful for disconnecting unwanted connections, that may pose a risk, as like Denial of Service, or which are likely to hamper the directory performance. For example:

  ```
  ldapexop -op unbind -dn cn=john
  ldapexop -op unbind -ip 9.182.173.43
  ldapexop -op unbind -dn cn=john -ip 9.182.173.43
  ldapexop -op unbind -all
  ```

► uniqueattr -a *<attributeType>*

  This extended operation is used to identify all the nonunique values for a particular attribute.

  -a *<attribute>*

  Specify the attribute for which all conflicting values are to be listed.

  > **Note:** Duplicate values for binary, operational, configuration attributes, and the objectclass attribute are not displayed. These attributes are not supported extended operations for unique attributes.

  For example:

  ```
  ldapexop -op uniqueattr -a "uid"
  ```

  The following line is added to the configuration file under the `cn=Directory,cn=RDBM Backends,cn=IBM Directory,cn=Schema,cn=Configuration` entry for this extended operation:

  ```
  ibm-slapdPlugin:extendedop /bin/libback-rdbm.dll initUniqueAttr
  ```

> **Note:** If no DN arguments are provided, the ldapexop command waits to read a list of DNs from standard input. To break out of the wait, use Ctrl+C or Ctrl+D.

SSL, TLS notes

The SSL or TLS - related functions associated with this utility are as like the ones described with ldapchangepwd above.

Diagnostics

Exit status is 0 if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error.

User type and user roles for extended operations

The following are the users and their roles for extended operations.

► User

Root administrator: This user is an administrative user whose "simple and External" with "SSL or TLS" bind credentials are stored under the cn=Configuration entry. This user's Kerberos bind credentials (optional) are stored under the cn=Kerberos,cn=Configuration entry. This user's Digest-MD5 bind credentials (optional) are stored under the cn=Digest,cn=Configuration entry. In addition, this type of user can bind to the Admin Daemon.

► Role(s)

– Server configuration administrator

This user has unrestricted access to all information in the configuration backend and can start/stop the server. The user can issue dynamic configuration updates.

– Directory administrator

This user has unrestricted access to directory data outside the configuration backend (schema, and RDBM backends). This user can search for one or two attributes in the configuration backend. This user may not have any authority to the operating specific backends (OS/400 system projection backend, z/OS RACF SDBM).

– Administrative group member

This user is basically an administrative user whose "simple or External" with "SSL or TLS, Kerberos (optional), and Digest-MD5 (optional)" credentials are stored under an entry in the subtree cn=Admingroup,cn=Configuration. In addition, this type of user can bind to the Admin Daemon.

► Role(s)

– Server configuration group member

This user has access to all configuration information except the administrator and admin group credentials. This user has the ability to start and stop the server. The user does not have the ability to add or remove members from the administrative group. The user is not be able to modify the DN, password, Kerberos ID, or Digest-MD5 ID of any administrative group member entry under

cn=AdminGroup,cn=Configuration. If the user is an Administrative Group Member the user is able to modify his own password, but is not able to modify his own DN, Kerberos ID, or Digest-MD5 ID. This user is also not able to see the password of any other administrative group member or the IBM Tivoli Directory Server administrator. In addition, this user is not able to add, delete, or modify the audit log settings (the entire cn=Audit,cn=Configuration entry) or clear the audit log. The user is not able to add or delete the cn=Kerberos,cn=Configuration or cn=Digest,cn=Configuration entries, but is able to search all attributes under these entries. The user is able to modify all attributes under these entries except the Kerberos and Digest-MD5 root administrator bind attributes. These users are not able to search or modify the ibm-slapdAdminDN, ibm-slapdAdminGroupEnabled or ibm-slapdAdminPW attributes under the cn=Configuration entry. The user can issue dynamic configuration updates.

- Directory administrator

  This user has unrestricted access to directory data outside the configuration backend (schema, and RDBM backends). This user can search for one or two attributes in the configuration backend. This user may not have any authority to the operating specific backends (OS/400 system projection backend, z/OS RACF SDBM).

- LDAP user type

  This user is a regular LDAP user whose credentials are stored in the DIT of the LDAP Server. The user's "simple and external" with "SSL or TLS" bind DN is the DN of an entry in the DIT. The user's password is stored in the userpassword attribute of this entry.

► Role(s)

  LDAP User Role: A user having almost no access to the configuration backend. This user can search for one or two attributes in the configuration backend. The user's access to directory data (schema, and RDBM backends) is controlled by ACLs.

10.4 The ldapmodify and ldapadd commands

This client tool helps in modifying existing entries in the directory or adding new ones. `ldapmodify` is not used for modifying the RDN values of entries. There is a separate client tool for this which will be explained in "The ldapmodrdn command" on page 270.

10.4.1 Synopsis

```
ldapmodify [-a] [-b] [-c] [-C charset] [-d debuglevel][-D binddn][-g] [-G
realm] [-h ldaphost] [-i file] [-k] [-K keyfile] [-m mechanism] [-M] [-N
certificatename] [-O maxhops] [-p ldapport] [-P keyfilepw] [-r] [-R] [-U
username] [-v] [-V] [-w passwd | ?] [-y proxydn] [-Y] [-Z]

ldapadd [-a] [-b] [-c] [-C charset] [-d debuglevel][-D binddn][-g] [-G realm]
[-h ldaphost] [-i file] [-k] [-K keyfile] [-m mechanism] [-M] [-N
certificatename] [-O maxhops] [-p ldapport] [-P keyfilepw] [-r] [-R] [-U
username] [-v] [-V] [-w passwd | ?] [-y proxydn] [-Y] [-Z]
```

10.4.2 Description

`ldapmodify` is a command-line interface to the `ldap_modify` and `ldap_add` library calls. `ldapadd` is implemented as a renamed version of `ldapmodify`. When invoked as ldapadd, the -a (add new entry) flag is turned on automatically.

`ldapmodify` opens a connection to an LDAP server, and binds to the server. You can use `ldapmodify` to modify or add entries. The entry information is read from standard input or from file through the use of the -i option.

To display syntax help for `ldapmodify` or `ldapadd`, type:

```
ldapmodify -?
```

Or:

```
ldapadd -?
```

10.4.3 Options

Options like -C *charset*, -c, -d *debuglevel*, -D *binddn*, -G *realm*, -h *ldaphost*, -K *keyfile*, -k, -m *mechanism*, -M, -N *certificatename*, -O *maxhops*, -p *ldapport*, -P *keyfilepw*, -R, -U *username*, -v, -V, -w *passwd | ?*, -y *proxydn*, -Y and -Z have already been discussed for the **ldapchangepwd** command in 10.1.2, "Options" on page 239.

Here are the options that are additional:

► -a

  Add new entries. The default action for `ldapmodify` is to modify existing entries. If invoked as `ldapadd`, this flag is always set.

► -b

  Assume that any values that start with a '/' are binary values and that the actual value is in a file whose path is specified in place of the value.

- ► -g

  Specifies not to strip the trailing spaces on attribute values.

- ► -i *file*

  Read the entry modification information from an LDIF file instead of from standard input. If an LDIF file is not specified, you must use standard input to specify the update records in LDIF format.

- ► -r

  Replace existing values by default.

Input format

The contents of the file (or the standard input if no -i flag is given on the command line) should conform to the LDIF format.

Alternative input format

An alternative input format is supported for compatibility with older versions of ldapmodify. This format consists of one or more entries separated by blank lines, where each entry looks like the following:

```
Distinguished Name (DN)
attr=attrvalue
[attr=attrvalue ...]
```

Where *attr* is the name of the attribute and value is the attrvalue.

By default, values are added. If the **-r** command line flag is given, the default is to replace existing values with the new one. It is permissible for a given attribute to appear more than once, for example, to add more than one value for an attribute. Also note that you can use a trailing '\\' to continue values across lines and preserve new lines in the value itself.

attr should be preceded by a **-** to remove a value. The **=** and value should be omitted to remove an entire attribute.

attr should be preceded by a **+** to add a value in the presence of the -r flag.

10.4.4 Examples

Lets see a set of examples illustrating the options/arguments that we have just discussed above.

Example 1

Assuming that the file /tmp/entrymods exists and has the following contents:

```
dn: cn=Modify Me, o=University of Higher Learning, c=US
changetype: modify
replace: mail
mail: modme@student.of.life.edu
-
add: title
title: Grand Poobah
-
add: jpegPhoto
jpegPhoto: /tmp/modme.jpeg
-
delete: description
-
```

The command:

```
ldapmodify -b -r -i /tmp/entrymods
```

will replace the contents of the Modify Me entry's mail attribute with the value modme@student.of.life.edu, add a title of Grand Poobah, and the contents of the file /tmp/modme.jpeg as a jpegPhoto, and completely remove the description attribute. These same modifications can be performed using the older ldapmodify input format, by modifying the file /tmp/entrymods as:

```
cn=Modify Me, o=University of Higher Learning, c=US
mail=modme@student.of.life.edu
+title=Grand Poobah
+jpegPhoto=/tmp/modme.jpeg
-description
```

And by using the command:

```
ldapmodify -b -r -i /tmp/entrymods
```

Example 2

Assuming that the file /tmp/newentry exists and has the following contents:

```
dn: cn=John Doe, o=University of Higher Learning, c=US
objectClass: person
cn: John Doe
cn: Johnny
sn: Doe
title: the world's most famous mythical person
mail: johndoe@student.of.life.edu
uid: jdoe
```

The command:

```
ldapadd -i /tmp/entrymods
```

adds a new entry for John Doe, using the values from the file /tmp/newentry.

Example 3

Assuming that the file /tmp/newentry exists and has the contents:

```
dn: cn=John Doe, o=University of Higher Learning, c=US
changetype: delete
```

The command:

```
ldapmodify -i /tmp/entrymods
```

removes John Doe's entry.

Example 4

Assuming that the file /tmp/newentry exists and has the contents:

```
dn: cn=Modify Me, o=University of Higher Learning, c=US
changetype: modify
replace: description
description:abc
```

The command:

```
ldapmodify -g -i /tmp/entrymods
```

retains the trailing spaces in the description field, that is, `abc`, as they are entered. It would be difficult to search for the spaces here! You may want to try this example in your environments to see that the trailing spaces are actually maintained.

Note: If no DN arguments are provided, the ldapmodify command waits to read a list of DNs from standard input. To break out of the wait, use Ctrl+C or Ctrl+D.

10.4.5 SSL, TLS notes

The SSL or TLS - related functions associated with this utility are as like the ones described in "The ldapchangepwd command" on page 239.

10.4.6 Diagnostics

Exit status is 0 if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error.

10.5 The ldapmodrdn command

This tool is specifically designed to modify the RDN part of an entry's dn.

10.5.1 Synopsis

```
ldapmodrdn [-c] [-C charset] [-d debuglevel][-D binddn] [-G realm] [-h
ldaphost] [-i file] [-k] [-K keyfile] [-m mechanism] [-M] [-n] [-N
certificatename] [-O hopcount] [-p ldapport] [-P keyfilepw] [-r] [-R] [-U
username] [-v] [-V] [-w passwd | ?] [-y proxydn] [-Y] [-Z] [dn newrdn | [-i
file]]
```

10.5.2 Description

`ldapmodrdn` is a command-line interface to the ldap_modrdn library call.

`ldapmodrdn` opens a connection to an LDAP server, binds, and modifies the RDN of entries. The entry information is read from standard input, from file through the use of the -f option, or from the command-line pair of dn and RDN.

Refer to "LDAP distinguished name syntax (DNs)" on page 43 for information about RDNs and DNs.

To display syntax help for `ldapmodrdn`, type:

```
ldapmodrdn -?
```

10.5.3 Options

The options like -c, -C *charset*, -d *debuglevel*, -D *binddn*, -G *realm*, -h *ldaphost*, -k, -K *keyfile*, -m *mechanism*, -M, -n, -N *certificatename,* -O *hopcount*, -p *ldapport*, -P *keyfilepw*, -R, -U *username*, -v, -V, -w *passwd | ?*, -y *proxydn*, -Y, -Z are already explained or would be explained in one of the other sections. Hence we are not taking them in this section. The illustrations can be applied to ldapmodrdn as was to the earlier client utilities.

► -i *file*

Read the entry modification information from the file instead of from standard input or the command-line (by specifying ran and newrdn). Standard input can be supplied from a file, as well using redirection ("< file").

► -r

  Remove old RDN values from the entry. Default action is to keep old values.

► dn *newrdn*

  See the following section, "Input format for dn newrdn" for more information.

Input format for dn newrdn

If the command-line arguments dn and newrdn are given, newrdn replaces the RDN of the entry specified by the DN, dn. Otherwise, the contents of file (or standard input if no - i flag is given) consist of one or more entries:

```
Distinguished Name (DN)
Relative Distinguished Name (RDN)
```

One or more blank lines may be used to separate each DN and RDN pair.

10.5.4 Examples

Here are a few examples of using the `ldapmodrdn` command.

Example 1

Assuming that the file /tmp/entrymods exists and has the contents:

```
cn=user, o=ibm, c=US
cn=NewUser
```

Note the output of the command:

```
C:\>ldapmodrdn -D cn=root -w secret  -i test.ldif
copying cn=user, o=ibm, c=US to cn=NewUser
```

Example 2

Assuming that the file /tmp/entrymods exists and has the contents:

```
cn=NewUser, o=ibm, c=US
cn=user
```

Observe the output of the command:

```
C:\>ldapmodrdn -D cn=root -w secret -r -i test.ldif
moving cn=NewUser, o=ibm, c=US to cn=user
```

In both examples the RDN is changed. It is changed from user to NewUser in example 1 and NewUser to user in example 2. However the thing to note is that by using -r we are moving the current value to a new one with an RDN change and in case we do not use the -r option we are copying the contents from one dn to another. Thus using -r is supposed to yield better modrdn performance.

> **Note:** If no DN arguments are provided, the ldapmodrn command waits to read a list of DNs from standard input. To break out of the wait, use Ctrl+C or Ctrl+D.

10.5.5 SSL, TLS notes

The SSL or TLS - related functions associated with this utility are as like the ones described with ldapchangepwd in "The ldapchangepwd command" on page 239.

10.5.6 Diagnostics

Exit status is 0 if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error.

10.6 The ldapsearch command

This is the most widely used client tool. The obvious reason being that the LDAP protocol is a read-optimization protocol and `ldapsearch` is a tool for reading/fetching data from the LDAP server.

10.6.1 Synopsis

```
ldapsearch [-a deref] [-A] [-b searchbase] [-B] [-C charset] [-d debuglevel]
[-D binddn] [-F sep] [-G realm] [-h ldaphost] [-i file] [-K keyfile] [-l
timelimit] [-L] [-m mechanism] [-M] [-n] [-N certificatename] [-o attr_type]
[-O maxhops] [-p ldapport] [-P keyfilepw] [-q pagesize] [-R] [-s scope ] [-t]
[-T seconds] [-U username] [-v] [-V version] [-w passwd | ?] [-z sizelimit] [-y
proxydn] [-Y] [-Z] filter [-9 p] [-9 s] [attrs...]
```

10.6.2 Description

`ldapsearch` is a command-line interface to the ldap_search library call.

`ldapsearch` opens a connection to an LDAP server, binds, and performs a search using the filter. The filter should conform to the string representation for LDAP filters (see `ldap_search` in the *IBM Tivoli Directory Server Version 5.2 C-Client SDK Programming Reference* for more information on filters).

You can get this document at:

http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html

If `ldapsearch` finds one or more entries, the attributes specified by **a**ttrs are retrieved and the entries and values are printed to standard output. If no attrs are listed, all attributes are returned.

To display syntax help for `ldapsearch`, type `ldapsearch -?`.

10.6.3 Options

The options like -C *charset*, -d *debuglevel*, -D *binddn*, -e, -G *realm*, -h *ldaphost*, -K *keyfile*, -m *mechanism*, -n, -O *maxhops*, -p *ldapport*, -P *keyfilepw*, -U *username*, -w *passwd* / *?*, -Y, -Z have already been discussed in 10.1, "The ldapchangepwd command" on page 239. The options that are specific to the **ldapsearch** command are:

► -a *deref*

Specify how aliases dereferencing is done. *deref* should be one of:

   – never: Aliases are never dereferenced.
   – always: Aliases are always dereferenced.
   – search: Aliases are deferenced when searching.
   – find: Aliases are dereferenced only when locating the base object.

► -A

Retrieve attributes only (no values). This is useful when you just want to see if an attribute is present in an entry and are not interested in the specific values.

► -b *searchbase*

Use *searchbase* as the starting point for the search instead of the default. If -b is not specified, this utility will examine the LDAP_BASEDN environment variable for a searchbase definition. If neither is set, the default base is set to "", which is a null search. A null search returns all the entries in the entire Directory Information Tree (DIT). This search requires a -s subtree option. Otherwise, an error message is displayed. Be aware that null based search requests consume a lot of resource.

► -B

Do not suppress display of non-ASCII values. This is useful when dealing with values that appear in alternate character sets such as ISO-8859.1. This option is implied by the -L option.

► -F *sep*

Use *sep* as the field separator between attribute names and values. The default separator is '=', unless the -L flag has been specified, in which case this option is ignored.

- -i *file*

  Read a series of lines from a file, performing one LDAP search for each line. In this case, the filter given on the command line is treated as a pattern where the first occurrence of %s is replaced with a line from file. If file is a single "-" character, then the lines are read from standard input.

  For example, in the command,

  ```
  ldapsearch -V3 -v -b "o=ibm,c=us" -D "cn=admin" -w ldap -i filter.input
  %s dn
  ```

  The file filter.input file might contain the following filter information:

  ```
  (cn=*Z)
  (cn=*Z*)
  (cn=Z*)
  (cn=*Z*)
  (cn~=A)
  (cn>=A)
  (cn<=B)
  ```

  Note: Each filter must be specified on a separate line.

  The command performs a search of the subtree o=ibm,c=us for each of the filters beginning with cn=*Z. When that search is completed, the search begins for the next filter cn=*Z* and so forth until the search for the last filter cn<=B is completed.

  Note: The -i < file> option replaces the -f < file> option. The -f option is still supported, although it is deprecated.

- -l *timelimit*

  Wait at most timelimit seconds for a search to complete.

- -L

  Display search results in LDIF format. This option also turns on the -B option, and causes the -F option to be ignored.

- -M

  Manage referral objects as regular entries.

- -N *certificatename*

  Specify the label associated with the client certificate in the key database file.

> **Note:** If the LDAP server is configured to perform server authentication only, a client certificate is not required. If the LDAP server is configured to perform client and server Authentication, a client certificate might be required. certificatename is not required if a default certificate/private key pair has been designated as the default. Similarly, certificatename is not required if there is a single certificate/private key pair in the designated key database file. This parameter is ignored if neither -Z nor -K is specified.

► -o *attr_type*

To specify an attribute to use for sort criteria of search results, you can use the -o (order) parameter. You can use multiple -o parameters to further define the sort order. In the following example, the search results are sorted first by surname (sn), then by given name, with the given name (givenname) being sorted in reverse (descending) order as specified by the prefixed minus sign (-):

```
-o sn -o -givenname
```

Thus, the syntax of the sort parameter is as follows:

[-]<attribute name>[:<matching rule OID>]

Where:

– attribute name is the name of the attribute you want to sort by.

– matching rule OID is the optional OID of a matching rule that you want to use for sorting.

– The minus sign (-) indicates that the results must be sorted in reverse order.

– The criticality is always critical.

The default ldapsearch operation is not to sort the returned results.

► -q *pagesize*

To specify paging of search results, two new parameters can be used: -q (query page size), and -T (time between searches in seconds).

In the following example, the search results return a page (25 entries) at a time, every 15 seconds, until all the results for that search are returned. The ldapsearch client handles all connection continuation for each paged results requested for the life of the search operation.

```
-q 25 -T 15
```

If the -v (verbose) parameter is specified, ldapsearch lists how many entries have been returned so far, after each page of entries returned from the server, for example, 30 total entries have been returned.

Multiple -q parameters are enabled such that you can specify different page sizes throughout the life of a single search operation. In the following example, the first page is 15 entries, the second page is 20 entries, and the third parameter ends the paged result/search operation:

```
-q 15 -q 20 -q 0
```

In the following example, the first page is 15 entries, and all the rest of the pages are 20 entries, continuing with the last specified -q value until the search operation completes:

```
-q 15 -q 20
```

The default ldapsearch operation is to return all the entries in a single request.

No paging is done for the default ldapsearch operation.

► -R

Specifies that referrals are not to be automatically followed.

► -s *scope*

Specify the *scope* of the search. scope should be one of:

– base - Search is limited to the base.

– one - Search is limited to one-level below the base and does not include the base.

– sub - Search covers the base as well as its descendants.

Note: If you specify a null search, either by not specifying a -b option or specifying -b "", you must the -s option. The default scope is disabled for a null search.

► -t

Write retrieved values to a set of temporary files. This is useful for dealing with non-ASCII values such as jpegPhoto or audio.

► -T *seconds*

Time between searches (in seconds). The **-T** option is only supported when the -q option is specified.

► -y *proxydn*

Specifies the DN to be used for proxied authorization. The earlier sections did not illustrate this feature. We will have an example in the Examples section down below on the use of this option.

► -z *sizelimit*

Limit the results of the search to at the most sizelimit entries. This makes it possible to place an upper bound on the number of entries that are returned for a search operation.

► -9 *p*

Sets criticality for paging to false. The search is handled without paging.

Here is an excerpt from the ITDS 52 Administration guide, which guides us for setting/unsetting this option

The LDAP server returns all referrals to the client at the end of a search request, the same as a search without any controls. That means that if the server has 10 pages of results returned, all the referrals are returned on the 10th page, not at the end of each page. When chasing referrals, the client application needs to send in an initial paged results request, with the cookie set to null, to each of the referral servers. It is up to the application using the client services to decide whether or not to set the criticality as to the support of paged results, and to handle a lack of support of this control on referral servers as appropriate based on the application. Additionally, the LDAP server does not ensure that the referral server supports paged results controls. Multiple lists could be returned to the client application, some not paged. It is at the client application's decision as to how to best present this information to the end user. Possible solutions include: Combine all referral results before presenting to the end user; show multiple lists and the corresponding referral server host name; take no extra steps and show all results to the end user as they are returned from the server. The client application must turn off referrals to get one truly paged list, otherwise when chasing referrals with the paged results search control specified, unpredictable results might occur.

► -9 *s*

Sets criticality for sorting to false. The search is handled without sorting.

Here is an excerpt from the ITDS 52 Administration guide, which guides on the setting/unsetting of this option:

The LDAP server returns all referrals to the client at the end of a search request. It is up to the application using the client services to decide whether to set the criticality of the sorted search request, and to handle a lack of support of those controls on referral servers as appropriate based on the application. Additionally, the LDAP server does not ensure that the referral server supports the sorted search control. Multiple lists could be returned to the client application, some not sorted. It is the client application's decision as to how best to present this information to the end user. Possible solutions include: combine all referral results before presenting to the end user; show multiple lists and the corresponding referral server host name; take no extra

steps and show all results to the end user as they are returned from the server. The client application must turn off referrals to get one truly sorted list, otherwise when chasing referrals with sorted search controls specified, unpredictable results might occur.

For more information on the paging/sorting criticality issues you may refer the ITDS 5.2 Administration Guide. The link for the same is provided in the earlier sections.

► filter

Specifies a string representation of the filter to apply in the search. Simple filters can be specified as attributetype=attributevalue. More complex filters are specified using a prefix notation according to the following Backus Naur Form (BNF)

```
<filter> ::='('<filtercomp>')'
<filtercomp> ::= <and>|<or>|<not>|<simple>
<and> ::= '&' <filterlist>
<or> ::= '|' <filterlist>
<not> ::= '!' <filter>
<filterlist> ::= <filter>|<filter><filtertype>
<simple> ::= <attributetype><filtertype><attributevalue>
<filtertype> ::= '='|'~='|'<='|'>='
```

The '~=' construct is used to specify approximate matching. The representation for <attributetype> and <attributevalue> are as described in "RFC 2252, LDAP V3 Attribute Syntax Definitions". In addition, <attributevalue> can be a single * to achieve an attribute existence test, or can contain text and asterisks (*) interspersed to achieve substring matching.

For example, the filter "mail=*"" finds any entries that have a mail attribute.The filter "mail=*@student.of.life.edu" finds any entries that have a mail attribute ending in the specified string. To put parentheses in a filter, escape them with a backslash (\) character.

> **Note:** A filter like "cn=Bob *", where there is a space between Bob and the asterisk (*), matches "Bob Carter" but not "Bobby Carter" in IBM Directory. The space between "Bob" and the wildcard character (*) affects the outcome of a search using filters.

Please refer RFC 2254, "A String Representation of LDAP Search Filters" for a more complete description of allowable filters.

Output format

If one or more entries are found, each entry is written to standard output in the form:

```
Distinguished Name (DN)
attributename=value
attributename=value
attributename=value
...
```

Multiple entries are separated with a single blank line. If the -F option is used to specify a separator character, it will be used instead of the '=' character. If the -t option is used, the name of a temporary file is used in place of the actual value. If the -A option is given, only the "attributename" part is written.

10.6.4 Examples

Here are some examples of the `ldapsearch` command.

Example 1

The command:

```
ldapsearch "cn=john doe" cn telephoneNumber
```

Performs a subtree search (using the default search base) for entries with a commonName of "john doe". The commonName and telephoneNumber values is retrieved and printed to standard output. The output might look something like this, if two entries are found:

```
cn=John E Doe, ou="College of Literature, Science, and the Arts",
ou=Students, ou=People, o=University of Higher Learning, c=US
cn=John Doe
cn=John Edward Doe
cn=John E Doe 1
cn=John E Doe
telephoneNumber=+1 313 555-5432

cn=John B Doe, ou=Information Technology Division,
ou=Faculty and Staff, ou=People, o=University of Higher Learning, c=US
cn=John Doe
cn=John B Doe 1
cn=John B Doe
telephoneNumber=+1 313 555-1111
```

Example 2

The command:

```
ldapsearch -t "uid=jed" jpegPhoto audio
```

This performs a subtree search using the default search base for entries with user id of "jed". The jpegPhoto and audio values are retrieved and written to temporary files. The output might look like this, if one entry with one value for each of the requested attributes is found:

```
cn=John E Doe, ou=Information Technology Division,ou=Faculty and
Staff,ou=People, o=University of Higher Learning, c=US
audio=/tmp/ldapsearch-audio-a19924
jpegPhoto=/tmp/ldapsearch-jpegPhoto-a19924
```

Example 3

This command:

```
ldapsearch -L -s one -b "c=US" "o=university*" o description
```

This will perform a one-level search at the c=US level for all organizations whose organizationName begins with university. Search results will be displayed in the LDIF format (You may refer the LDAP Data Interchange Format in the *ITDS 52 Administration Guide* for the detailed specifics on LDIF format. The link is provided in the earlier sections). The organizationName and description attribute values will be retrieved and printed to standard output, resulting in output similar to this:

```
dn: o=University of Alaska Fairbanks, c=US
o: University of Alaska Fairbanks
description: Preparing Alaska for a brave new tomorrow
description: leaf node only

dn: o=University of Colorado at Boulder, c=US
o: University of Colorado at Boulder
description: No personnel information
description: Institution of education and research

dn: o=University of Colorado at Denver, c=US
o: University of Colorado at Denver
o: UCD
o: CU/Denver
o: CU-Denver
description: Institute for Higher Learning and Research

dn: o=University of Florida, c=US
o: University of Florida
o: UFl
description: Shaper of young minds
```

Example 4

This command:

```
ldapsearch -b "c=US" -o ibm-slapdDN "objectclass=person" ibm-slapdDN
```

This performs a subtree level search at the c=US level for all persons. When this special attribute is used for sorted searches, the search results are sorted by the string representation of the Distinguished Name (DN). The output might look something like this:

```
cn=Al Edwards,ou=Widget Division,ou=Austin,o=IBM,c=US
cn=Al Garcia,ou=Home Entertainment,ou=Austin,o=IBM,c=US
cn=Amy Nguyen,ou=In Flight Systems,ou=Austin,o=IBM,c=US
cn=Arthur Edwards,ou=Widget Division,ou=Austin,o=IBM,c=US
cn=Becky Garcia,ou=In Flight Systems,ou=Austin,o=IBM,c=US
cn=Ben Catu,ou=In Flight Systems,ou=Austin,o=IBM,c=US
cn=Ben Garcia Jr,ou=Home Entertainment,ou=Austin,o=IBM,c=US
cn=Bill Keller Jr.,ou=In Flight Systems,ou=Austin,o=IBM,c=US
cn=Bob Campbell,ou=In Flight Systems,ou=Austin,o=IBM,c=US
```

Example 5

Here is an example to see the behavior of the -M and -R flags, pertaining to referrals. Assumption is that there exists a custom object class, which inherits from the referral class and also has the userPassword attribute in it. The custom object class is named myreferral and cn=myref,o=ibm,c=us is an object of the same. cn=myref,o=ibm,c=us refers to cn=user1,o=ibm,c=us.

```
C:\>ldapsearch -D cn=myref,o=ibm,c=us -w user1 -s base -b
cn=myref,o=ibm,c=us objectclass=*
cn=user1,o=ibm,c=us
objectclass=organizationalPerson
objectclass=person
objectclass=top
sn=1
cn=user1

C:\>ldapsearch -D cn=myref,o=ibm,c=us -w user1 -M -s base -b
cn=myref,o=ibm,c=us objectclass=*
ldap_simple_bind: Invalid credentials

C:\>ldapsearch -D cn=myref,o=ibm,c=us -w myref -M -s base -b
cn=myref,o=ibm,c=us objectclass=*
cn=myref,o=ibm,c=us
objectclass=myreferral
objectclass=referral
objectclass=top
ref=ldap://localhost/cn=user1,o=ibm,c=us
cn=myref

C:\>ldapsearch -D cn=myref,o=ibm,c=us -w myref -R -s base -b
cn=myref,o=ibm,c=us objectclass=*
ldap_simple_bind: Referral returned
```

```
C:\>ldapsearch -D cn=root -w secret -R -s base -b cn=myref,o=ibm,c=us
objectclass=*
ldap_search: Referral returned
Unfollowed referral: ldap://localhost/cn=user1,o=ibm,c=us
```

As shown above, if you try to chase a referral with the binddn as the dn of the
referral and the bind password as that of the target (cn=user1,o=ibm,c=us) you
do reach there. However, if you treat cn=myref,o=ibm,c=us as a normal entry
(-M) then the bind password of the referral is expected and not of the target
object. And lastly if the referrals aren't chased (-R) then you get back a referral
from the server, which is displayed only when bound as an administrator.

Example 5

Assuming the fact that cn=alias2,o=ibm,c=us is an alias of an entry
cn=alias1,o=ibm,c=us which in turn is an alias of cn=user1,o=ibm,c=us, here is
what We will get on the different options of dereferencing:

► deref: always

```
C:\>ldapsearch -D cn=root -w secret -a always -b cn=alias1,o=ibm,c=us
objectclass=* dn
cn=user1,o=ibm,c=us

C:\>ldapsearch -D cn=root -w secret -a always -b cn=alias2,o=ibm,c=us
objectclass=* dn
cn=user1,o=ibm,c=us
```

► deref: searching with searchbase as a non-alias

```
C:\>ldapsearch -D cn=root -w secret -a searching -b o=ibm,c=us
objectclass=* dn | grep -i alias
<No output>
```

► deref: finding with searchbase as a non-alias

```
C:\>ldapsearch -D cn=root -w secret -a finding -b o=ibm,c=us objectclass=*
dn | grep -i alias
cn=alias1,o=ibm,c=us
cn=alias2,o=ibm,c=us
```

► deref: searching with searchbase as a alias

```
C:\>ldapsearch -D cn=root -w secret -a searching -b cn=alias2,o=ibm,c=us
objectclass=* dn
cn=user1,o=ibm,c=us
```

► deref: finding with searchbase as a alias

```
C:\>ldapsearch -D cn=root -w secret -a finding -b cn=alias2,o=ibm,c=us
objectclass=* dn
cn=user1,o=ibm,c=us
```

► deref: never

```
C:\>ldapsearch -D cn=root -w secret -a never -b cn=alias2,o=ibm,c=us
objectclass=* dn
cn=alias2,o=ibm,c=us
```

The above examples clearly demonstrate the different ways the aliases can be treated by `ldapsearch`. Also, it is noteworthy that the difference in the output from -deref : searching and -deref : finding is based on the fact whether the searchbase is an alias or not.

Example 6

The following command shows the use of the -A option:

```
C:\>ldapsearch -D cn=root -w secret -a never -b cn=alias2,o=ibm,c=us -A
objectclass=*
cn=alias2,o=ibm,c=us
aliasedobjectname
objectclass
cn
```

Example 7

Here we bring out the difference between using the -B switch and not using it:

```
C:\>ldapsearch -D cn=root -w secret -a never -b cn=user,o=ibm,c=us -B
objectclass=* jpegPhoto
cn=user,o=ibm,c=US
jpegPhoto=BMµ?

C:\>ldapsearch -D cn=root -w secret -a never -b cn=user,o=ibm,c=us
objectclass=* jpegPhoto
cn=user,o=ibm,c=US
jpegPhoto=NOT ASCII
```

As shown above, if -B switch is used, the binary data will also appear in the ldapsearch output, though it is not apparently meaningful.

Example 8

This example brings out the difference between using the -F sep option and not using it:

```
C:\>ldapsearch -D cn=root -w secret -b cn=user,o=ibm,c=us objectclass=* sn
cn=user,o=ibm,c=US
sn=j
C:\>ldapsearch -D cn=root -w secret -b cn=user,o=ibm,c=us -F :
objectclass=* sncn=user,o=ibm,c=US
sn:j
```

Example 9

This example shows the use of the -l option.

Consider the following search:

```
ldapsearch -D cn=root -w secret -l 1 -b o=ibm,c=us objectclass=*
```

Currently o=ibm,c=us has 10,000 entries below it. However, the ldapsearch is given a total time limit of 1second for returning all the results. After that time exceeds and if ldapsearch has not finished with all the desired entries, the following message is flashed and the search stops:

```
ldap_search: Timelimit exceeded
```

Example 10

With the assumption that there exists an entry cn=user1,o=ibm,c=us with two children as cn=h1,cn=user1,o=ibm,c=us and cn=h2,cn=user1,o=ibm,c=us, this example shows the use of the -o option:

```
C:\>ldapsearch -D cn=root -w secret -o cn -s sub -b cn=user1,o=ibm,c=us
objectclass=* dn
cn=h1,cn=user1,o=ibm,c=us
cn=h2,cn=user1,o=ibm,c=us
cn=user1,o=ibm,c=us

C:\>ldapsearch -D cn=root -w secret -o -cn -s sub -b cn=user1,o=ibm,c=us
objectclass=* dn
cn=user1,o=ibm,c=us
cn=h2,cn=user1,o=ibm,c=us
cn=h1,cn=user1,o=ibm,c=us
```

Example 11

This example shows the differences between the various options associated with the parameter scope, specified using -s:

```
C:\>ldapsearch -D cn=root -w secret -s base -b cn=user1,o=ibm,c=us
objectclass=* dn
cn=user1,o=ibm,c=us

C:\>ldapsearch -D cn=root -w secret -s sub -b cn=user1,o=ibm,c=us
objectclass=* dn
cn=user1,o=ibm,c=us
cn=h1,cn=user1,o=ibm,c=us
cn=h2,cn=user1,o=ibm,c=us

C:\>ldapsearch -D cn=root -w secret -s one -b cn=user1,o=ibm,c=us
objectclass=* dn
cn=h1,cn=user1,o=ibm,c=us
cn=h2,cn=user1,o=ibm,c=us
```

Example 12

This example illustrates the use of the -z flag:

```
C:\>ldapsearch -D cn=root -w secret -b cn=user1,o=ibm,c=us objectclass=* dn
cn=user1,o=ibm,c=us
cn=h1,cn=user1,o=ibm,c=us
cn=h2,cn=user1,o=ibm,c=us

C:\>ldapsearch -D cn=root -w secret -b cn=user1,o=ibm,c=us -z 1
objectclass=* dn

cn=user1,o=ibm,c=us
ldap_search: Sizelimit exceeded
```

Example 13

This example illustrates the use of the proxy dn, through the -y flag.

Assuming that there exist 2 users cn=user1,o=ibm,c=us. user2 is not allowed to
see the password of user1, however user2 is in the Proxy group.

```
C:\>ldapsearch -D cn=user2,o=ibm,c=us -w user2 -s base -b
cn=user1,o=ibm,c=us objectclass=*
cn=user1,o=ibm,c=us
objectclass=organizationalPerson
objectclass=person
objectclass=top
sn=1
cn=user1

C:\>ldapsearch -D cn=user2,o=ibm,c=us -w user2 -y cn=user1,o=ibm,c=us -s
base -b cn=user1,o=ibm,c=us objectclass=*
cn=user1,o=ibm,c=us
objectclass=organizationalPerson
objectclass=person
objectclass=top
sn=1
cn=user1
userpassword={SHA}s9qne0wEqVUbh4HQMZH+CY8yXmc=
```

Note the difference between the two searches that are fired on the LDAP server
and also the difference in the output that are we seeing.

In the first case user2 is trying to fetch the entire entry of user1, by binding as
user2. The result is that user1 does not reveal the userPassword to user2.

Now in the second case, as user2 is in the Proxy Group, it is able to fire a query
on the entry of user1, as a proxy of user1 (using the -y option) and get the
userPassword.

For information on the Proxy Group, you may refer the ITDS v 52 Administration Guide. The link for the same is put up in one of the sections above.

10.6.5 SSL, TLS notes

The SSL or TLS - related functions associated with this utility are as like the ones described with `ldapchangepwd` in "The ldapchangepwd command" on page 239.

10.6.6 Diagnostics

Exit status is 0 if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error.

10.7 Summary

After seeing these utilities let us have a summary of what was done in this chapter. This chapter was mainly focussed on learning the following client utilities:

► `ldapchangepwd`

► `ldapdelete`

► `ldapexop`

► `ldapmodify` and `ldapadd`

► `ldapmodrdn`

► `ldapsearch`

We also saw the detailed explanations on the options that the above utilities would take.

11

Schema management

This chapter provides an introduction to the IBM Tivoli Directory Server for both distributed platforms and z/OS. Features and functions described in this chapter are based on ITDS 5.2 and LDAP for z/OS V1R4, therefore some of the functionality described may not be available in earlier releases. The topics covered in this section include:

► What is the schema
► Modifying the schema
► Migrating a schema
► Dynamic schema

11.1 What is the schema

A schema is a set of rules that governs the way that data can be stored in the directory. The schema defines the type of entries allowed, their attribute structure, and the syntax of the attributes.

Data is stored in the directory using directory entries. A entry consists of an object class, which is required, and its attributes. Attributes can be either required or optional. The object class specifies the kind of information that the entry describes and defines the set of attributes it contains. Each attribute has one or more associated values. See "Modifying the schema" on page 292 for additional information about entries.

The schema for the IBM Directory Version 5.2 is predefined, however, you can modify the schema, if you have additional requirements.

The IBM Tivoli Directory Server Version 5.2 includes dynamic schema support. The schema is published as part of the directory information, and is available in the Subschema entry (`DN="cn=schema"`).

The schema has more configuration information than that included in the LDAP Version 3 Request For Comments (RFCs) or standard specifications. For example, for a given attribute, you can state which indexes must be maintained. This additional configuration information is maintained in the subschema entry as appropriate. An additional object class is defined for the subschema entry IBMsubschema, which has "MAY" attributes that hold the extended schema information.

IBM Tivoli Directory Server requires that the schema defined for a naming context be stored in a special directory entry, `"cn=schema"`. The entry contains all of the schema defined for the server. To retrieve schema information, you can perform an `ldap_search` by using the following:

```
DN: "cn=schema", search scope: base, filter: objectclass=subschema
or objectclass=*
```

The schema provides values for the following attribute types:

► objectClasses
► attributeTypes
► IBMAttributeTypes
► matching rules
► LDAP syntaxes

The syntax of these schema definitions is based on the LDAP Version 3 RFCs. A sample schema can be seen in Example 11-1.

Example 11-1   Sample schema

```
objectclasses=( 1.3.6.1.4.1.1466.101.120.111
                NAME 'extensibleObject'
                SUP top AUXILIARY )

  objectclasses=(  2.5.20.1
                NAME 'subschema'
                AUXILIARY MAY
                    ( dITStructureRules
                    $ nameForms
                    $ ditContentRules
                    $ objectClasses
                    $ attributeTypes
                    $ matchingRules
                    $ matchingRuleUse  ) )
  objectclasses=( 2.5.6.1
                NAME 'alias'
                SUP top STRUCTURAL
                MUST aliasedObjectName )

  attributeTypes {
      ( 2.5.18.10 NAME 'subschemaSubentry' EQUALITY distinguishedNameMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 NO-USER-MODIFICATION
        SINGLE-VALUE USAGE directoryOperation )
      ( 2.5.21.5 NAME 'attributeTypes'
          EQUALITY objectIdentifierFirstComponentMatch
          SYNTAX 1.3.6.1.4.1.1466.115.121.1.3 USAGE directoryOperation )
      ( 2.5.21.6 NAME 'objectClasses'
          EQUALITY objectIdentifierFirstComponentMatch
          SYNTAX 1.3.6.1.4.1.1466.115.121.1.37 USAGE directoryOperation )
          SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE directoryOperation )
}


    ldapSyntaxes {
            ( 1.3.6.1.4.1.1466.115.121.1.5 DESC 'Binary' )
            ( 1.3.6.1.4.1.1466.115.121.1.7 DESC 'Boolean' )
            ( 1.3.6.1.4.1.1466.115.121.1.12 DESC 'DN' )
            ( 1.3.6.1.4.1.1466.115.121.1.15 DESC 'Directory String' )
            ( 1.3.6.1.4.1.1466.115.121.1.24 DESC 'Generalized Time' )
            ( 1.3.6.1.4.1.1466.115.121.1.26 DESC 'IA5 String' )
            ( 1.3.6.1.4.1.1466.115.121.1.27 DESC 'INTEGER' )
            ( 1.3.6.1.4.1.1466.115.121.1.50 DESC 'Telephone Number' )
            ( 1.3.6.1.4.1.1466.115.121.1.53 DESC 'UTC Time' )
    }
```

```
matchingRules {
    ( 2.5.13.2 NAME 'caseIgnoreMatch'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
    ( 2.5.13.0 NAME 'objectIdentifierMatch'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )
    ( 2.5.13.30 NAME 'objectIdentifierFirstComponentMatch'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )
    ( 2.5.13.4 NAME 'caseIgnoreSubstringsMatch'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.58 )
}
```

As shown in the preceding example, it is not required that all of the attribute values of a given attribute type be provided in a single production.

11.1.1 Available schema files

Several schema files are shipped with the Tivoli IBM Directory Server and the z/OS IBM Directory Server to be used as a base for a directory, to allow for product integration, and to provide an area for custom schema changes and additions.

Modifying schema files directly is not recommended. However, additional information concerning adding and modifying Schema objectclasses and attributes can be found in "Modifying the schema" on page 292.

Table 11-1 reviews the schema files that ship with the product.

Table 11-1   Schema files

File name	Description
V3.Schema.at	Schema attribute file, specific to the operation of IBM Directory Server. Includes configuration information, password policy enforcement, and replication.
V3.Schema.oc	Schema objectclass file, specific to the operation of IBM Directory Server. Includes configuration information, password policy enforcement, and replication.
V3.ibm.at	Schema attribute file for IBM related products and technologies. One example is the AIX Authentication schema.
V3.ibm.oc	Schema objectclass file for IBM related products and technologies. One example is the AIX Authentication schema.

File name	Description
V3.user.at	Industry standard LDAP schema including attributes for person, organizationalperson, and inetorgperson.
V3.user.oc	Industry standard LDAP schema including objectclasses for person, organizationalperson, and inetorgperson.
V3.modifiedschema	Custom schema additions should be placed in this file.

11.1.2 Schema support

The IBM Directory supports standard directory schema as defined in the following:

► The Internet Engineering Task Force (IETF) LDAP Version 3 RFCs, such as RFC 2252 and 2256

► The Directory Enabled Network (DEN)

► The Common Information Model (CIM) from the Desktop Management Task Force (DMTF)

► The Lightweight Internet Person Schema (LIPS) from the Network Application Consortium

IBM Tivoli Directory Server 5.2 includes the LDAP Version 3 defined schema in the default schema configuration. It also includes the DEN schema definitions as well as a set of extended common schema definitions that other IBM products share when they exploit the LDAP directory. They include:

► Objects for white page applications such as eperson, group, country, organization, organization unit and role, locality, state, and so forth

► Objects for other subsystems such as accounts, services and access points, authorization, authentication, security policy, and so forth

Note: z/OS LDAP schema extensions can be found in the /usr/lpp/ldap/etc folder.

11.1.3 OID

An object identifier (OID) is a string, of decimal numbers, that uniquely identifies an objectclass or attribute. An OID is required for all objectclasses and attributes that are defined in the LDAP directory. There are two ways to handle new OIDs. They can be found at the Internet Assigned Number Authority Web site, http://www.iana.org/iana/, or be generated through a text OID assignment. An OID can be assigned the value of the objectclass or attribute name appended

with an '-oid'. For example, a new attribute, myattribute, can be assigned the OID myattribute-oid.

11.1.4 Inheritance

IBM Tivoli Directory Server version 5.2 supports object inheritance for object class and attribute definitions. A new objectclass can be defined with parent classes (multiple inheritance) and the additional or changed attributes.Each entry is assigned to a single structural object class. All object classes inherit from the abstract object class top. They can also inherit from other object classes.

As already discussed, the structure of an objectclass determines the list of required and allowed attributes for a particular entry. An object class can only inherit from object classes that precede it. For example, the objectclass organizationalPerson is able to inherit the attributes of the person objectclass, automatically inheriting the required and permitted attributes for that objectclass as well as any additional permissions specific to organizationalPerson.

11.2 Modifying the schema

The schema for the IBM Tivoli Directory Server Version 5.2 is predefined, however, you can modify the schema, if necessary. It is a good idea to take a look at what the current schema initially looks like. this can be done with a simple `ldapsearch` command. The following command exports the schema to an LDIF file called schemaout.ldif.

```
ldapsearch -L -h <ipaddress> -p <port> -b "cn=schema, <suffix>"
objectclass=* > schemaout.ldif
```

It should be noted that editing objectclasses and attributes directly is not recommended. A more accommodating option is to utilize objectclass inheritance, creating a new objectclass which inherits all properties of the desired objectclass with customized attributes and modifications contained within its definition.

11.2.1 IBMAttributetypes

The IBMAttributeTypes attribute can be used to define schema information not covered by the LDAP Version 3 standard for attributes. In the command line examples below, each attribute added has an example of how an IBMAttributetype can be added/modified/etc along side initial actions. This is effective in allowing attributes to include indexing extensions. Below is a template that an IBMAttributeType must comply with grammatically:

```
IBMAttributeTypesDescription = "(" whsp
```

```
numericoid whsp
[ "DBNAME" qdescrs ] ; at most 2 names (table, column)
[ "ACCESS-CLASS" whsp IBMAccessClass whsp ]
[ "LENGTH" wlen whsp ] ; maximum length of attribute
[ "EQUALITY" [ IBMwlen ] whsp ] ; create index for matching rule
[ "ORDERING" [ IBMwlen ] whsp ] ; create index for matching rule
[ "APPROX" [ IBMwlen ] whsp ] ; create index for matching rule
[ "SUBSTR" [ IBMwlen ] whsp ] ; create index for matching rule
[ "REVERSE" [ IBMwlen ] whsp ] ; reverse index for substring
whsp ")"
```

11.2.2 Working with objectclasses

Working with objectclasses allows a user to customize his or her directory
beyond the base LDAP installation. Below are instructions for adding, modifying,
and deleting an objectclass. All instructions are command line based and may be
used with either the distributed LDAP or z/OS LDAP directory servers.

Adding an objectclass

To add an object class using the command line, issue the following command:

```
ldapmodify -D <adminDN> -w <adminPW> -i <filename>
```

Where *<filename>* contains:

```
dn: cn=Schema
changetype: modify
add: objectclasses
objectclasses: ( <myobjectClass-oid> NAME <myObjectClass> DESC <An
objectclass I defined for my LDAP application> SUP <objectclassinheritance>
<objectclasstype> MUST (<attribute1> $ <attribute2>) MAY (<attribute1> $
<attribute2>) )
```

Editing an objectclass

View the object classes contained in the schema issue the command:

```
ldapsearch -b cn=schema -s base objectclass=* objectclasses
```

To edit an object class using the command line, issue the following command:

```
ldapmodify -D <adminDN> -w <adminPW> -i <filename>
```

Where <filename> contains:

```
dn: cn=schema
changetype: modify
replace: objectclasses
objectclasses: ( <myobjectClass-oid> NAME '<myObjectClass>' DESC '<An
objectclass I defined for my LDAP application>' SUP
```

```
'<newsuperiorclassobject>' <newobjectclasstype> MUST (<attribute1> $
<attribute2>) MAY (<attribute1> $ <attribute2>) )
```

Deleting an objectclass

View the object classes contained in the schema issue the command:

```
ldapsearch -b cn=schema -s base objectclass=* objectclasses
```

Select the object class you want to delete and issue the following command:

```
ldapmodify -D <adminDN> -w <adminPW> -i <filename>
```

Where <filename> contains:

```
dn: cn=schema
changetype: modify
delete: objectclasses
objectclasses: ( <myobjectClass-oid> NAME '<myObjectClass>' DESC '<An
object class I defined for my LDAP application>' SUP
'<objectclassinheritance>' <objectclasstype > MUST (<attribute1> $
<attribute2>) > MAY (<attribute1> $ <attribute2>) )
```

11.2.3 Working with attributes

Every object class includes a number of required attributes and optional
attributes. Required attributes are the attributes that must be present in entries
using the object class. Optional attributes are the attributes that may be present
in entries using the object class.

Below are instructions for adding, modifying, and deleting an attribute. All
instructions are command line based and may be used with either the distributed
LDAP or z/OS LDAP directory servers.

Adding an attribute

The following example adds an attribute type definition for an attribute called
myattribute.

```
ldapmodify -D <admindn> -w <adminpw> -i myschema.ldif
```

Where the myschema.ldif file contains:

```
dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: ( myAttribute-oid NAME ( 'myAttribute' )
   DESC 'An attribute I defined for my LDAP application'
   EQUALITY 2.5.13.2 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
   USAGE userApplications )
-
```

```
    add: ibmattributetypes
    ibmattributetypes: ( myAttribute-oid DBNAME ( 'myAttrTable' 'myAttrColumn'
    )
        ACCESS-CLASS normal LENGTH 200 )
```

Editing an attribute

This example adds indexing to the attribute, so that searching on it is faster. Use
the **ldapmodify** command and the LDIF file to change the definition:

```
    ldapmodify -D <admindn> -w <adminpw> -i myschemachange.ldif
```

Where the myschemachange.ldif file contains:

```
    dn: cn=schema
    changetype: modify
    replace: attributetypes
    attributetypes: ( myAttribute-oid NAME ( 'myAttribute' ) DESC 'An attribute
    I defined for my LDAP application' EQUALITY 2.5.13.2 SYNTAX
    1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
    -
    replace: ibmattributetypes
    ibmattributetypes: ( myAttribute-oid DBNAME ( 'myAttrTable' 'myAttrColumn'
    ) ACCESS-CLASS normal LENGTH 200 EQUALITY SUBSTR )
```

Note: Both portions of the definition (attributetypes and ibmattributetypes)
must be included in the replace operation, even though only the
ibmattributetypes section is changing.

Deleting an attribute

This example deletes the attribute 'myattribute' from the directory schema.

```
    ldapmodify -D <admindn> -w <adminpw> -i myschemadelete.ldif
```

Where the myschemadelete.ldif file includes:

```
    dn: cn=schema
    changetype: modify
    delete: attributetypes
    attributetypes: ( myAttribute-oid NAME ( 'myAttribute' ) DESC 'An attribute
    I defined for my LDAP application' EQUALITY 2.5.13.2 SYNTAX
    1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
    -
    delete: ibmattributetypes
    ibmattributetypes: ( myAttribute-oid DBNAME ( 'myAttrTable' 'myAttrColumn'
    ) ACCESS-CLASS normal LENGTH 200 EQUALITY SUBSTR )
```

11.2.4 Disallowed schema changes

Not all schema changes are allowed.

Change restrictions include the following:

- ► Any change to the schema must leave the schema in a consistent state.
- ► An attribute type that is a supertype of another attribute type may not be deleted.
- ► An attribute type that is a MAY or a MUST attribute type of an object class may not be deleted.
- ► An object class that is a superclass of another may not be deleted.
- ► Attribute types or object classes that refer to nonexisting entities (for example, syntaxes or object classes) cannot be added.
- ► Attribute types or object classes cannot be modified in such a way that they end up referring to nonexisting entities (for example, syntaxes or object classes).

Changes to the schema that affect the operation of the server are also not allowed. The following schema definitions are required by the directory server. They must not be changed.

Object classes

The following object class definitions must not be modified:

- ► accessGroup
- ► accessRole
- ► alias
- ► referral
- ► replicaObject
- ► top

Attributes

The following attribute definitions must not be modified:

- ► Operational attributes
- ► Restricted attributes
- ► Root DSE attributes
- ► Schema definition attributes
- ► Configuration attributes
- ► User application attributes
- ► Syntaxes
- ► Matching rules

11.3 Indexing

Index rules attached to attributes make it possible to retrieve information faster. If only the attribute is given, no indexes are maintained. ITDS provides the following indexing rules:

- ► Equality
- ► Ordering
- ► Approximate
- ► Substring
- ► Reverse

Indexing rules specifications for attributes: Specifying an indexing rule for an attribute controls the creation and maintenance of special indexes on the attribute values. This greatly improves the response time to searches with filters which include those attributes. The five possible types of indexing rules are related to the operations applied in the search filter.

Equality

Applies to the following search operations:

```
equalityMatch '='
```

For example:

```
"cn = John Doe"
```

Ordering

Applies to the following search operation:

- ► greaterOrEqual '>='
- ► lessOrEqual '<='

For example:

```
"sn >= Doe"
```

Approximate

Applies to the following search operation:

```
approxMatch '~='
```

For example:

```
"sn ~= doe"
```

Substring

Applies to the search operation using the substring syntax:

```
substring '*'
```

For example:

```
"sn = McC*"
"cn = J*Doe"
```

Reverse

Applies to the following search operation:

```
'*' substring
```

For example:

```
"sn = *baugh"
```

At a minimum, it is recommended that you specify equal indexing on any attributes that are to be used in search filters.

11.4 Migrating the schema

Schema migrations are used most commonly when replicas servers are being set up, or additional LDAP environments are being created. These migrations allow the modifications made to a schema, adding, deleting, modifying any objectclasses or attributes, to be carried through to all new directories.

When IBM Tivoli Directory Server 5.2 is being used, the command line instructions should be followed. When z/OS LDAP is used, the `schema2ldif` utility can be used.

11.4.1 Exporting the schema

In this section we discuss exporting the schema.

From the command line

The following command will dump the existing schema into a file called `schemaout` in LDIF format.

```
ldapsearch -h <ipaddress> -L -D  cn=<admindn> -w <password> -b
"cn=schema,<suffix>" objectclass=* > schemaout.ldif
```

Schema2LDIF utility

The `schema2ldif` utility offered by the z/OS platform is used primarily to migrate from an RDBM or SDBM schema syntax to a schema syntax that will integrate well with the TDBM backend. This utility takes .at and .oc files containing attributes and objectclasses and outputs an ldif files that can be used with the **`ldapmodifyschema2ldif -s at.in oc.in -d output.ldif -l log.out`** command to be loaded into a TDBM backend LDAP directory.

The `schema2ldif` file can be found in the folder /usr/lpp/ldap/bin. In the following example, the **`schema2ldif`** command is using two input files, at.in and oc.in, and creating one ldif file as an output file. The log.out file is the file in which all activity is logged for this utility. For additional information, including additional options for this utility, see the z/OS LDAP Server Administration and Use Guide.

```
schema2ldif -s at.in oc.in -d output.ldif -l log.out
```

11.4.2 Importing the schema

Both methods of exporting the schema will result in an LDIF file. This LDIF file can then be used by the following LDAP command to load the schema into the directory.

```
ldapmodify -h ldaphost -p ldapport -D adminDN -w passwd -f schemaout.ldif
```

11.5 Dynamic schema

The IBM Tivoli Directory Server Version 5.2 includes dynamic schema support. To perform a dynamic schema change, use the `ldap_modify` API with a DN of `cn=schema`. Only one dynamic change to a schema entry can be made at a time.

To delete a schema entity, provide the oid in parentheses: (oid).

To add or replace a schema entity, you MUST provide a LDAP Version 3 definition and you MAY provide the IBM definition. In all cases, you must provide only the definition or definitions of the schema entity that you want to affect.

For example, to delete the attribute type 'cn' (its OID is 2.5.4.3), use `ldap_modify()` with:

```
LDAPMod attr;
LDAPMod *attrs[] = { &attr, NULL };
char *vals [] = { "( 2.5.4.3 )", NULL };
attr.mod_op = LDAP_MOD_DELETE;
attr.mod_type = "attributeTypes";
attr.mod_values = vals;
ldap_modify_s(ldap_session_handle, "cn=schema", attrs);
```

To add a new attribute type bar with OID 20.20.20 that has a NAME of length 20chars:

```
char *vals1[] = { "( 20.20.20 NAME fbarf SUP NAME )", NULL };
char *vals2[] = { "( 20.20.20 LENGTH 20 )", NULL };
LDAPMod attr1;
LDAPMod attr2;
LDAPMod *attrs[] = { &attr1, &attr2, NULL };
attr1.mod_op = LDAP_MOD_ADD;
attr1.mod_type = "attributeTypes";
attr1.mod_values = vals1;
attr2.mod_op = LDAP_MOD_ADD;
attr2.mod_type = "IBMattributeTypes";
attr2.mod_values = vals2;
ldap_modify_s(ldap_session_handle, "cn=schema", attrs);
```

Note: You cannot change the ACCESS-CLASS type to or from system or restricted.

Note: Dynamic schema changes can be performed only by a replication supplier or the administrator DN. When a dynamic schema change is performed, it is replicated just like any other ldap_modify operation.

12

Group and role management

This chapter discusses the way groups and roles are managed in the directory server. The groups/roles are the easiest means of segregating users with like profiles/privileges. For example, if you want to give a common permission to a set of 100 users then it would be difficult if you try manipulating access on an individual basis. On the other hand if you create a group with these 100 users as its members and assign the permissions you want to the group, things would be much more easier. On same lines you can create a role and have that role assigned to these 100 users. You can set the permissions at the role level and achieve the same thing as you do by using a group.

There is a very thin boundary between a role and a group. There is just a conceptual difference between the two. Even the objectclass used to define a a nd a Role are same. The Role is just a different view of a Group and vise-versa. You say a group of engineers. Now it is also possible to create a Role with the name engineer. Whoever conforms to this Role possesses the attributes assigned to the Role. In other words, whoever is assigned the role of *system administrator* possesses the privileges assigned to the role system administrator. It is as like saying whoever belongs to the group of system administrators would be carrying the attributes defined for the role system administrators.

12.1 Groups

A group is a list, a collection of names. A groups can be used in attributes as like `aclentry, ibm-fliterAclEntry`, and `entryowner` to implement access control or in application-specific uses such as a mailing list (Refer Chapter 14, "Access control" on page 395, for further information on aclentry, ibm-filterAclEntry and entryowner attributes). Groups can be defined as either static, dynamic, or nested.

12.1.1 Static groups

A static group defines each member individually using the structural objectclass `groupOfNames, groupOfUniqueNames, accessGroup`, or `accessRole`, or by using the auxiliary objectclass `ibm-staticgroup`. The objectclass `groupOfNames` has its `member` attribute as a required attribute. The objectclass `groupOfUniqueNames` has its `uniqueMember` attribute as a required attribute. The objectclasses `accessGroup`, `accessRole` and `ibm-staticgroup` have their attribute `member` as an optional one and hence can be empty/null (with no members) while creating the group. Note that attributes `member` and `uniqueMember` attributes are multivalued.

A typical group entry is:

```
dn: cn=Dev.Staff,ou=Austin,c=US
objectclass: accessGroup
cn: Dev.Staff
member: cn=John Doe,o=IBM,c=US
member: cn=Jane Smith,o=IBM,c=US
member: cn=James Smith,o=IBM,c=US
```

Each group object contains a multivalued attribute consisting of member DNs.

Upon deletion of an access group, the access group is also deleted from all ACLs to which it has been applied.

Let us see how a static group is created. There are two ways of doing the same.

Using Web Administration Tool
Using the Web Administration Tool:

1. Connect to the desired server using the Web Administration Tool.

2. If you have not done so already, expand the **Directory management** category in the navigation area.

3. Click **Add an entry**.

4. Select one of the **Structural object classes** from the list box. You may select any of the four objectclasses: `groupOfNames, groupOfUniqueNames,`

`accessGroup,` or `accessRole`. You also have an option to select any other objectclass in this step and go to the next page.

5. If you have already specified one of the above four groups as your objectclass then you may skip this step, else you can select the auxiliary class `ibm-staticGroup` and press **Next**.

6. In the Relative DN field, enter the relative distinguished name (RDN) of the group that you are creating, for example, `cn=staticgroup`.

7. In the Parent DN field, enter the distinguished name of the tree entry you selected, for example, `o=IBM, c=US`. You can also click **Browse** to select the **Parent DN** from the list. You can also expand the selection to view other choices lower in the subtree. Specify your choice and click **Select** to specify the Parent DN that you want. The Parent DN defaults to the entry selected in the tree.

> **Note:** If you started this task from the Manage entries panel, this field is prefilled for you. You selected the Parent DN before clicking Add to start the process of creating a group.

8. Now you would need to specify the member, if member is a mandatory field (depends upon the objectclass you have chosen for your static group). If you do not want to add members at this stage you may skip the step (this is possible only in case the member/uniquemember attribute is optional).

9. The group is created at this point and you can view it as any other normal entry.

The above procedure shows that creating a group is as like creating a normal entry. The only point to remember is that you need to use the objectclasses as applicable for groups, those mentioned above.

Through the command line

Here is the procedure of creating a static group using an LDIF file.

Assuming that we have an ldif file, test.ldif with the following entries:

```
dn: cn=staticGroup 1,o=ibm,c=us
objectclass: accessGroup
objectclass: top
cn: staticGroup 1
```

The following command would add the group for you:

```
C:\> ldapadd -D <admin DN> -w <admin PW> -i test.ldif
adding new entry cn=staticGroup 1,o=ibm,c=us
```

You can verify that the group actually exists or not, by browsing through the Web Administration tool or by firing a search as like:

```
E:\>ldapsearch -D <admin DN> -w <admin PW> -b "cn=staticgroup 1,o=ibm,c=us"
objectclass=*
cn=staticGroup 1,o=ibm,c=us
objectclass=accessGroup
objectclass=top
cn=staticGroup 1
```

Now let us see the procedure for adding/removing a member to/from a group.

Using the Web Administration Tool

To use this:

1. If you have not done so already, expand the **Directory management** category in the navigation area, then click **Manage entries**.

2. You can expand the various subtrees and select the entry (**group**) that you want to work on. Click **Edit attributes** from the right-side tool bar.

3. At the Required attributes tab enter the values for the required attributes.

4. Search for the member attribute. Depending upon the objectclass, chosen for the group, it may be in the tab of Required attributes or in the tab of Optional attributes.

5. To add members to the group:

   a. Click either **Multiple values** which is by the Member attribute field on the Required attributes tab, or click **Members** by the Member attribute field on the Members tab.

   b. In the Member field, enter the DN of the entry you want to add.

   c. Click **Update/Add** (depends upon the panel your in to add the member).

   d. Click **OK**.

6. To remove members from the group:

   a. Either click Multiple values by the Members attribute field on either the **Required attributes** or the **Other attributes** tabs.

   b. Select the entry you want to remove.

   c. Click **Remove**.

   d. Click **OK**.

Refer to Figure 12-1 on page 305 for the screenshot of the panel where you would be adding/removing members to/from a group.
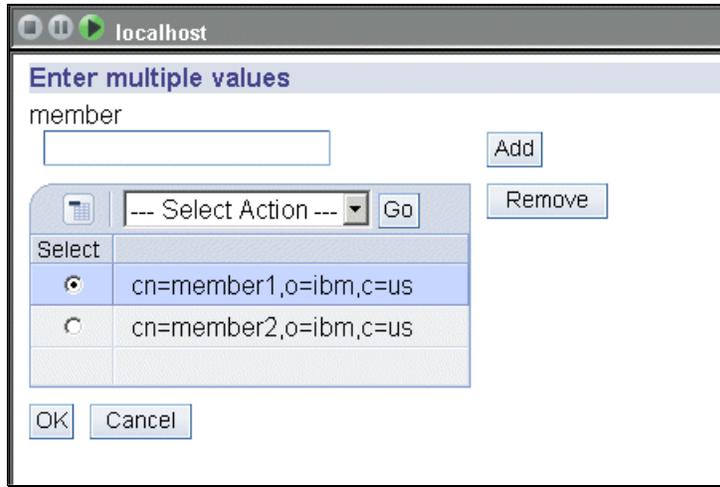
Figure 12-1   Add or remove members

Through command line

This syntax is not much different than the ldapmodify syntax you use to modify an entry.

Adding a member

Assuming the fact that we have an ldif file, test.ldif, with the following contents:

```
dn: cn=staticGroup 1,o=ibm,c=us
changetype: modify
add: member
member: cn=member4,o=ibm,c=us
```

Here is the command you need to execute:

```
E:\>ldapmodify -D cn=root -w secret -f test.ldif
modifying entry cn=staticGroup 1,o=ibm,c=us
```

Here is the verification that the member actually got added:

```
E:\>ldapsearch -D cn=root -w secret -b "cn=staticgroup 1,o=ibm,c=us"
objectclass=* ibm-allmembers
cn=staticGroup 1,o=ibm,c=us
ibm-allmembers=CN=USER1,O=IBM,C=US
ibm-allmembers=CN=MEMBER4,O=IBM,C=US
```

Do not worry much about the attribute used to fetch the members of the group. You will be seeing more details on the same, shortly.

Removing a member

Assuming the fact that we have an ldif file, test.ldif, with the following contents:

```
dn: cn=staticGroup 1,o=ibm,c=us
changetype: modify
add: member
member: cn=member4,o=ibm,c=us
```

Here is the command you need to execute:

```
E:\>ldapmodify -D cn=root -w secret -f test.ldif
modifying entry cn=staticGroup 1,o=ibm,c=us
```

Here is the procedure you need to use to verify that the member actually got removed:

```
E:\>ldapsearch -D cn=root -w secret -b "cn=staticgroup 1,o=ibm,c=us"
objectclass=* ibm-allmembers
cn=staticGroup 1,o=ibm,c=us
ibm-allmembers=CN=USER1,O=IBM,C=US
```

12.1.2 Dynamic groups

A dynamic group defines its members differently than a static group. Instead of listing them individually, the dynamic group defines its members using an LDAP search. The dynamic group uses the structural objectclass `groupOfURLs` (or auxiliary objectclass `ibm-dynamicGroup`) and the attribute, `memberURL` to define the search using a simplified LDAP URL syntax.

```
ldap:///<base DN of search>? ?<scope of search>?<searchfilter>
```

Note: From the LDAP URL that is shown above it is clear, that the host name must not be present in the syntax. The remaining parameters are just like normal ldap URL syntax. Each parameter field must be separated by a ?, even if no parameter is specified. Normally, a list of attributes to return would be included between the base DN and scope of the search. This parameter is also not used by the server when determining dynamic membership, and so may be omitted, however, the separator ? must still be present.

Where:

► base DN of search

This is the point from which the search begins in the directory. It can be the suffix or root of the directory such as ou=Austin. This parameter is required.

► scope of search

Specifies the extent of the search. The default *scope* is base.

– base - Returns information only about the base DN specified in the URL.

– one - Returns information about entries one level below the base DN specified in the URL. It does not include the base entry.

– sub - Returns information about entries at all levels below and includes the base DN.

► searchfilter

This is the filter that you want to apply to the entries within the scope of the search. See the ldapsearch filter option in Chapter 10 for information about the syntax of the searchfilter. The default is objectclass=*.

The search for dynamic members is always internal to the server, so unlike a full ldap URL, a host name and port number is never specified, and the protocol is always ldap (never ldaps). The memberURL attribute may contain any kind of URL, but the server only uses memberURLs beginning with ldap:/// to determine dynamic membership.

Examples

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

```
ldap:///cn=John Doe, cn=Employees, o=Acme, c=US
```

All entries that are 1-level below cn=Employees, and the filter defaults to objectclass=*:

```
ldap:///cn=Employees, o=Acme, c=US??one
```

All entries that are under o=Acme,c=us with the objectclass=person:

```
ldap:///o=Acme, c=US??sub?objectclass=person
```

Depending on the object classes you use to define user entries, those entries might not contain attributes which are appropriate for determining group membership. You can use the auxiliary object class, ibm-dynamicMember, to extend your user entries to include the ibm-group attribute. This attribute allows you to add arbitrary values to your user entries to serve as targets for the filters of your dynamic groups. For example, the members of this dynamic group are entries directly under the cn=users,ou=Austin entry that have an ibm-group attribute of GROUP1:

```
dn: cn=GROUP1,ou=Austin
objectclass: groupOfURLs
cn: GROUP1
memberURL: ldap:///cn=users,ou=Austin??one?(ibm-group=GROUP1)
```

Here is an example member of cn=GROUP1,ou=Austin:

```
dn: cn=Group 1 member, cn=users, ou=austin
objectclass: person
objectclass: ibm-dynamicMember
sn: member
userpassword: memberpassword
ibm-group: GROUP1
```

Now let us see the procedure for adding/removing a member to/from a group.

Using the Web Administration Tool

The procedure for adding a dynamic group through the Web Administration tool is as like the procedure for static groups, except for the fact that:

► You need to use the structural object class groupOfUrls or either of the auxiliary classes ibm-dynamicGroup or ibm-dynamicMember for creating the dynamic group.

► You need to mention the LDAP URL for the dynamic members against the memberURL attribute, in case the objectclass, chosen for the group is either groupOfUrls or ibm-dynamicGroup.

► In case you have selected ibm-dynamicMember as your auxiliary objectclass, you need to use the attribute ibm-group as explained earlier.

► Both the memberURL and the ibm-group are multivalued optional attributes.

Refer to Figure 12-2 on page 309 for a screenshot of the panel listing the users of a dynamic group, based on the LDAP URL you have specified. Please note that a similar output is expected to verify that the dynamic group creation was successful.

Figure 12-2   Members evaluated against an LDAP URL

Now let us see the similar procedure via command line.

Using the command line

Assuming that we have an ldif file named test.ldif as like:

```
cn=dynamicgroup 1,o=ibm,c=us
objectclass=groupOfURLs
objectclass=top
cn=dynamicgroup 1
memberurl=ldap:///o=ibm,c=us??one?cn=user*
```

Execute the following command to add the group:

```
E:\>ldapadd -D cn=root -w secret -f test.ldif
adding new entry cn=dynamicgroup 1,o=ibm,c=us
```

Let us see what are the members, that the group contains:

```
E:\>ldapsearch -D cn=root -w secret -b "cn=dynamicgroup 1,o=ibm,c=us"
objectclass=* ibm-allmembers
cn=dynamicgroup 1,o=ibm,c=us
```

```
ibm-allmembers=CN=USER1,O=IBM,C=US
ibm-allmembers=CN=USER3,O=IBM,C=US
ibm-allmembers=CN=USER4,O=IBM,C=US
```

Deleting or removing members is very much like normal entries. You would just need to modify the relevant attributes that hold the URL to the group members. You can do this either through the Web Administration Tool or through the command line.

12.1.3 Nested groups

The nesting of groups enables the creation of hierarchical relationships that can be used to define inherited group membership. A nested group is defined as a child group entry whose DN is referenced by an attribute contained within a parent group entry. A parent group is created by extending one of the structural group object classes (`groupOfNames`, `groupOfUniqueNames`, `accessGroup`, `accessRole`, or `groupOfURLs`) with the addition of the `ibm-nestedGroup` auxiliary object class. After nested group extension, zero or more `ibm-memberGroup` attributes may be added, with their values set to the DNs of nested child groups. For example:

```
dn: cn=Group 2, cn=Groups, o=IBM, c=US
objectclass: groupOfNames
objectclass: ibm-nestedGroup
objectclass: top
cn: Group 2
description: Group composed of static, and nested members.
member: cn=Person 2.1, cn=Dept 2, cn=Employees, o=IBM, c=US
member: cn=Person 2.2, cn=Dept 2, cn=Employees, o=IBM, c=US
ibm-memberGroup: cn=Group 8, cn=Nested Static, cn=Groups, o=IBM, c=US
```

Note: The introduction of cycles into the nested group hierarchy is not allowed. If it is determined that a nested group operation results in a cyclical reference, either directly or through inheritance, it is considered a constraint violation and therefore, the update to the entry fails.

Now let us see the procedure for adding/removing a member to/from a group.

Using the Web Administration Tool
The procedure for adding a dynamic group through the Web Administration tool remains the same as the static group, except for the fact that:

► You need to have one structural objectclass as like the one for static groups / dynamic groups.

► You need to have one auxiliary objectclass, which happens to be ibm-nestedGroup.

- The rules for mentioning the static members/dynamic members remain the same as explained earlier.
- The nested group needs to be mentioned against the ibm-memberGroup attribute, which happens to be a multi-valued, optional attribute.

Refer Figure 12-3 for a screenshot of the panel listing the users of a nested group, based on the static members and the member group(s) that you have specified. Please note that a similar output is expected to verify that the nested group creation was successful.

Figure 12-3   Member listing of a nested group

The procedure on the command line can be derived on the lines of command line procedures for static and dynamic groups, it is just a matter of using the right objectclass, while defining the group.

12.1.4 Hybrid groups

Any of the structural group object classes, mentioned in the earlier sections, can be extended such that group membership is described by a combination of static, dynamic, and nested member types.

For example:

```
dn: cn=Group 10, cn=Groups, o=IBM, c=US
objectclass: groupOfURLs
objectclass: ibm-nestedGroup
objectclass: ibm-staticGroup
objectclass: top
cn: Group 10
description: Group composed of static, dynamic, and nested members.
memberURL: ldap:///cn=Austin, cn=Employees, o=IBM,
c=US??one?objectClass=person
ibm-memberGroup: cn=Group 9, cn=Nested Dynamic, cn=Groups, o=IBM, c=US
member: cn=Person 10.1, cn=Dept 2, cn=Employees, o=IBM, c=US
member: cn=Person 10.2, cn=Dept 2, cn=Employees, o=IBM, c=US
```

The methods to create such a group are just the combination of what we have seen in case of static groups, dynamic groups and nested groups as three separate cases.

12.1.5 Determining group membership

Two operational attributes can be used to query aggregate group membership. For a given group entry, the `ibm-allMembers` operational attribute enumerates the aggregate 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` operational attribute enumerates the aggregate set of groups, including ancestor groups, to which that user has membership.

A requester may only receive a subset of the total data requested, depending on how the ACLs have been set on the data. Anyone can request the ibm-allMembers and ibm-allGroups operational attributes, but the data set returned only contains data for the LDAP entries and attributes that the requester has access rights to. The user requesting the ibm-allMembers or ibm-allGroups attribute must have access to the member or uniquemember attribute values for the group and nested groups in order to see static members, and must be able to perform the searches specified in the memberURL attribute values in order to see dynamic members. Let us take some examples. First let us see a hierarchy tree, which is going to be used in our examples.
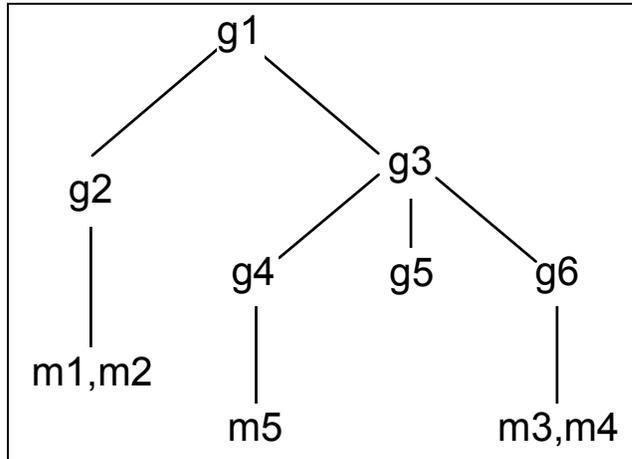
Figure 12-4   Hierarchy of groups and members

Hierarchy examples

For this example as shown in Figure 12-4, m1 and m2 are in the member attribute of g2. The ACLs for g2 allows user1 to read the member attribute, but user2 does not have access to the member attribute. The entry LDIF for the g2 entry is as follows:

```
dn: cn=g2,cn=groups,o=ibm,c=us
objectclass: accessGroup
cn: g2
member: cn=m1,cn=users,o=ibm,c=us
member: cn=m2,cn=users,o=ibm,c=us
aclentry: access-id:cn=user1,cn=users,o=ibm,c=us:normal:rsc
aclentry:
access-id:cn=user2,cn=users,o=ibm,c=us:normal:rsc:at.member:deny:rsc
```

The g4 entry uses the default aclentry, which allows both user1 and user2 to read its member attribute. The LDIF for the g4 entry is as follows:

```
dn: cn=g4,cn=groups,o=ibm,c=us
objectclass: accessGroup
cn: g4
member: cn=m5, cn=users,o=ibm,c=us
```

The g5 entry is a dynamic group, which gets its two members from the memberURL attribute. The LDIF for the g5 entry is as follows:

```
dn: cn=g5, cn=groups,o=ibm,c=us
objectclass: container
objectclass: ibm-dynamicGroup
cn: g5
```

```
memberURL: ldap:///cn=users,o=ibm,c=us??sub?(|(cn=m3)(cn=m4))
```

The entries m3 and m4 are members of group g5 because they match the memberURL attribute. The ACL for the m3 entry allows both user1 and user2 to search for it. The ACL for the m4 entries does not allow user2 to search for it. The LDIF for m4 is as follows:

```
dn: cn=m4, cn=users,o=ibm,c=us
objectclass:person
cn: m4
sn: four
aclentry: access-id:cn=user1,cn=users,o=ibm,c=us:normal:rsc
aclentry: access-id:cn=user2,cn=users,o=ibm,c=us
```

Example 1

User1 does a search to get all the members of group g1. User1 has access to all members, so they are all returned.

```
ldapsearch -D cn=user1,cn=users,o=ibm,c=us -w user1pwd -s base -b cn=g1,
cn=groups,o=ibm,c=us objectclass=* ibm-allmembers
   cn=g1,cn=groups,o=ibm,c=us
   ibm-allmembers: CN=M1,CN=USERS,O=IBM,C=US
   ibm-allmembers: CN=M2,CN=USERS,O=IBM,C=US
   ibm-allmembers: CN=M3,CN=USERS,O=IBM,C=US
   ibm-allmembers: CN=M4,CN=USERS,O=IBM,C=US
   ibm-allmembers: CN=M5,CN=USERS,O=IBM,C=US
```

Example 2

User2 does a search to get all the members of group g1. User2 does not have access to members m1 or m2 because they do not have access to the member attribute for group g2. User2 has access to the member attribute for g4 and therefore has access to member m5. User2 can perform the search in the group g5 memberURL for entry m3, so that members are listed, but cannot perform the search for m4.

```
ldapsearch -D cn=user2,cn=users,o=ibm,c=us -w user2pwd -s base -b cn=g1,
cn=groups,o=ibm,c=us objectclass=* ibm-allmembers
   cn=g1,cn=groups,o=ibm,c=us
   ibm-allmembers: CN=M3,CN=USERS,O=IBM,C=US
   ibm-allmembers: CN=M5,CN=USERS,O=IBM,C=US
```

Example 3

User2 does a search to see if m3 is a member of group g1. User2 has access to do this search, so the search shows that m3 is a member of group g1.

```
ldapsearch -D cn=user2,cn=users,o=ibm,c=us -w user2pwd -s base -b cn=m3,
cn=users,o=ibm,c=us objectclass=* ibm-allgroups
   cn=m3,cn=users,o=ibm,c=us
   ibm-allgroups: CN=G1,CN=GROUPS,O=IBM,C=US
```

Example 4

User2 does a search to see if m1 is a member of group g1. User2 does not have access to the member attribute, so the search does not show that m1 is a member of group g1.

```
ldapsearch -D cn=user2,cn=users,o=ibm,c=us -w user2pwd -s base -b
cn=m1,cn=users,o=ibm,c=us objectclass=* ibm-allgroups
    cn=m1,cn=users,o=ibm,c=us
```

Checking group membership using the Web Administration Tool

To check:

1. Connect to the relevant server through Web Administration Tool.

2. If you have not done so already, expand the **Directory management** category in the navigation area.

3. Click **Manage entries**.

4. Select a user from the directory tree and click the Edit attributes icon that appears at the right side of the main panel.

5. Click the **Memberships** tab.

6. That will show you all the groups to which this user/entry belongs to and also a lot of additional stuff as is explained further.

Refer to Figure 12-5 for the panel which shows the group memberships of a user.
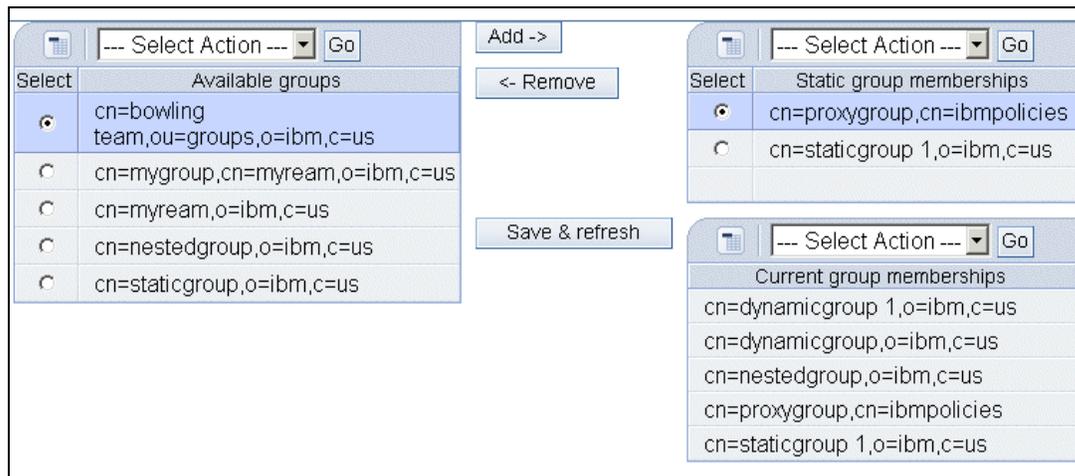
Figure 12-5   Change group membership

To modify the memberships for the user, the Change memberships panel (as shown in Figure 12-5 on page 315) displays the Available groups to which the user can be added, as well as the entry's Static Group Memberships.

1. Select a group from Available groups and click **Add** to make the entry a member of the selected group.

2. Select a group from Static Group Memberships and click **Remove** in case you need to remove the entry from the selected group.

3. Click **OK** to save your changes or click **Cancel** to return to the previous panel without saving your changes.

12.1.6 Group object classes

In this section we discuss group object classes.

ibm-dynamicGroup

This auxiliary class allows the optional `memberURL` attribute. Use it with a structural class such as `groupOfNames` to create a hybrid group with both static and dynamic members.

ibm-dynamicMember

This auxiliary class allows the optional `ibm-group` attribute. Use it as a filter attribute for dynamic groups.

ibm-nestedGroup

This auxiliary class allows the optional `ibm-memberGroup` attribute. Use it with a structural class such as `groupOfNames` to enable sub-groups to be nested within the parent group.

ibm-staticGroup

This auxiliary class allows the optional `member` attribute. Use it with a structural class such as `groupOfURLs` to create a hybrid group with both static and dynamic members.

> **Note:** The `ibm-staticGroup` is the only class for which `member` is optional, all other classes taking member require at least 1 `member`.

12.1.7 Group attribute types

In this section we discuss group attribute types.

ibm-allGroups

Shows all groups to which an entry belongs. An entry can be a member directly by the `member`, `uniqueMember`, or `memberURL` attributes, or indirectly by the `ibm-memberGroup` attribute. This Read-only operational attribute is not allowed in a search filter.

ibm-allMembers

Shows all members of a group. An entry can be a member directly by the `member`, `uniqueMember`, or `memberURL` attributes, or indirectly by the `ibm-memberGroup` attribute. This Read-only operational attribute is not allowed in a search filter.

ibm-group

This is an attribute taken by the auxiliary class `ibm-dynamicMember`. Use it to define arbitrary values to control membership of the entry in dynamic groups. For example, add the value "Bowling Team" to include the entry in any `memberURL` that has the filter "ibm-group=Bowling Team".

ibm-memberGroup

This is an attribute taken by the auxiliary class `ibm-nestedGroup`. It identifies sub-groups of a parent group entry. Members of all such sub-groups are considered members of the parent group when processing ACLs or the `ibm-allMembers` and `ibm-allGroups` operational attributes. The sub-group entries themselves are not members. Nested membership is recursive.

12.2 Roles

Role-based authorization is a conceptual complement to the group-based authorization, and is useful in some cases. As a member of a role, you have the authority to do what is needed for the role in order to accomplish a job. Unlike a group, a role comes with an implicit set of permissions. There is not any built-in assumption about what permissions are gained (or lost) by being a member of a group.

Roles are similar to groups in that they are represented in the directory by an object. Additionally, roles contain a group of DNs. Roles which are to be used in access control must have an objectclass of `AccessRole`. The `Accessrole` objectclass is a subclass of the `GroupOfNames` objectclass.

For example, if there is a collection of DNs such as 'sys admin', your first reaction may be to think of them as the 'sys admin group' (since groups and users are the most familiar types of privilege attributes). However, since there are a set of

permissions that you would expect to receive as a member of 'sys admin' the collection of DNs may be more accurately defined as the 'sys admin role'.

12.3 Summary

Now let us go through what we have covered in this chapter:

► We have seen that the directory management becomes much easier by means of groups and roles.

► We have seen the different types of groups:

– Static groups

– Dynamic groups

– Nested groups

– Hybrid groups

► We have seen how we can determine whether a given user is a member of a specific group and vice-versa.

► We have seen the objectclasses and attributes pertaining to groups.

► We have seen a set of examples to explain the concept and implementation of groups and roles.

13

Replication

Replication is the technique of duplicating data between multiple directories for performance, scalability and redundancy. It is a way to bring multiple geographic areas together into one enterprise directory.

These multiple copies are kept in sync with one or more main directory server called *Supplier* or what most call a master or writable server and one or more *Consumers* or what most call replica or read only servers.

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.

Replication provides a directory user with two main advantages:

► Reliability: The consumers act as backup copies of the suppliers and data can be restored from the consumer in case all data is lost from the supplier due to some catastrophic failure. With multiple masters you will also have a writable backup to take over if the first master goes down.

► Performance: Client search requests can be distributed across the Consumers instead of the single Supplier, thereby reducing the response time of the Supplier and hence increasing performance. With multiple master you will also be able to distribute applications to different masters to offload the writable work on having only one master.

13.1 General replication concepts

This section defines the general replication concepts.

13.1.1 Terminology

IBM Directory 4.1 and earlier supported a master-replica replication model. There are three types of directories for this: masters, replicas, and peers.

1. Master: The master server contains the master directory information from where updates are propagated to the replicas. All changes are made and occur on the master server, and the master is responsible for propagating these changes to the replicas.

2. Replica: An additional server that contains a directory replica. The replicas must be exact copies of the master. The replica provides a backup to the master server. Even if the master server crashes, or is unreadable, the replica can still fulfill search requests and provide access to the data. A replica can be promoted to a master if the master will no longer be available.

3. Peer to Peer: Peer replication is a replication in which multiple servers are masters. However, unlike a multi-master environment. no conflict resolution is done among peer servers. LDAP servers accept the updates provided by peer servers, and update their own copies of the data. No consideration is given for the order the updates are received, or whether multiple updates conflict.

With IBM Directory 5.1 and later there have been a number of changes and improvements. The master/replica model was changed to a Supplier/Consumer model. The following identify the supplier/consumer replication methods available:

1. Cascading replication: A replication topology in which there are multiple tiers of servers. A peer/master server replicates to a set of read-only (forwarding) servers which in turn replicate to other servers. Such a topology off-loads replication work from the master servers.

2. Consumer server: A server which receives changes through replication from another (supplier) server.

3. Credentials: Identifies the method and required information that the supplier uses in binding to the consumer. For simple binds, this includes the DN and password. The credentials are stored in an entry and the DN of this entry is specified in the replica agreement.

4. Forwarding server: A read-only server that replicates all changes sent to it. This contrasts to a peer/master server in that it is read only and it can have no peers.

5. Gateway server: A server that forwards all replication traffic from the local replication site where it resides to other Gateway servers in the replicating network. Also receives replication traffic from other Gateway servers within the replication network, which it forwards to all servers on its local replication site. Gateway servers must be masters (writable).

6. Master server: A server which is writable (can be updated) for a given subtree.

7. Nested subtree: A subtree within a replicated subtree of the directory.

8. Peer server: The term used for a master server when there are multiple masters for a given subtree. A peer server does not replicate changes sent to it from another peer server; it only replicates changes that are originally made on it or are received from another client not bound as the Master Server DN.

9. Replica group: The first entry created under a replication context has objectclass *ibm-replicaGroup* and represents a collection of servers participating in replication. It provides a convenient location to set ACLs to protect the replication topology information. The administration tools currently support one replica group under each replication context, named ibm-replicagroup=default.

10. Replica subentry: Below a replica group entry, one or more entries with objectclass *ibm-replicaSubentry* may be created; one for each server participating in replication as a supplier. The replica subentry identifies the role the server plays in replication: master or read-only. A read-only server might, in turn, have replication agreements to support cascading replication.

11. Replicated subtree: A portion of the DIT that is replicated from one server to another. Under this design, a given subtree can be replicated to some servers and not to others. A subtree can be writable on a given server, while other subtree's may be read-only.

12. Replicating network: A network that contains connected replication sites.

13. Replication agreement: Information contained in the directory that defines the 'connection' or 'replication path' between two servers. One server is called the supplier (the one that sends the changes) and the other is the consumer (the one that receives the changes). The agreement contains all the information needed for making a connection from the supplier to the consumer and scheduling replication.

14. Replication context: Identifies the root of a replicated subtree. The *ibm-replicationContext* auxiliary object class may be added to an entry to mark it as the root of a replicated area. The configuration information related to replication is maintained in a set of entries created below a replication context.

15. Replication site: A Gateway server and any master, peer or replica servers configured to replicate together.

16. Schedule: Replication can be scheduled to occur at particular times, with changes on the supplier accumulated and sent in a batch. The replica agreement contains the DN for the entry that supplies the schedule.

17. Supplier server: A server which sends changes to another (consumer) server.

13.1.2 How replication functions

Specific entries (suffixes or non-suffix entries) in the directory are identified as the roots of replicated subtrees by adding the `ibm-replicationContext` auxiliary class to them. Any entry added below this root, as its direct or indirect child, will be replicated to the respective replica servers as defined in the replication agreement. The replicated subtree continues down the DIT until it reaches another entry with the `ibm-replicationContext` auxiliary class. At this point the former replicated subtree ends and new replicated subtree begins. All the replicated subtrees get independently replicated and may get replicated to the same or different replicas using the same or different credentials.

All replicated roots have a replica group entry created directly below them. It gives a collective view of server participating in replication for the particular replicated root as all replica subentries and agreements are created below it.

Every server that acts as a supplier has a corresponding ibm-replicaSubentry entry created below the replica group. This entry contains the mandatory server-id of the supplier server among other things.

Each replica server is represented by an `ibm-replicationAgreement` entry created below the `ibm-replicaSubentry`. The replication agreement represents a path from the supplier to the consumer. If we have an `ibm-replicationAgreement` entry for server B created directly below the ibm-replicaSubentry for server A, then B is replica of A for the given subtree. Hence, any changes made to the given replicated root/subtree at server A will be propagated to server B.

Any updates directed against a read-only replica are automatically redirected to the corresponding supplier which then sends the updates to the replica.

Figure 13-1 on page 323 shows the hierarchy of the above mentioned objectclasses in the DIT.

Note: While adding the objectclasses, additional information as required for the objectclasses should be entered. Two replicas have been shown in the figure, but any number of replicas can be configured for a given supplier.

```
      o=ibm,c=us ((Replicated Subtree with ibm-replicationContext auxilliary class)
         |
         |
      ibm-replicaGroup=default  (replica group entry)
         |
      ibm-replicaSubentry  (this represents the supplier server)
         |
                ibm-replicationAgreement (for first replica)
         |
         └────ibm-replicationAgreement (for second replica)
```
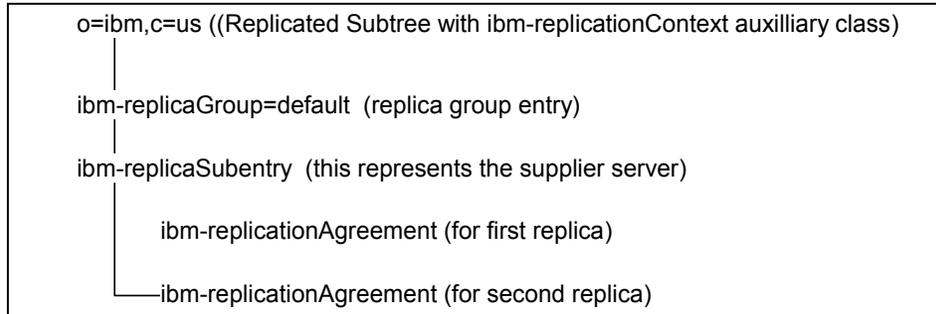
Figure 13-1   Hierarchy of the different object classes required in replication

Starting with IBM Directory Server 5.1, the roles played by a directory server in a replication topology no more apply to the entire server (DIT) but to a particular subtree in the Server (DIT). This means, a given directory server can be a Master / Peer (Supplier) with respect to one subtree and a replica (Consumer) with respect to another subtree at the same time.

Another change that was made to the IBM Directory starting with 5.1 was the adding of ibm-slapd Server Id. This is a single value attribute that identifies the server for use in replication. By default when you install the directory for the first time a UUID type number is generated and assigned to that directory. This is used anytime you need to identify a server in the replication. It can be found in the ibmslapd.conf file under dn: cn=Configuration

```
ibm-slapdServerId: 1320e26e-3457-4e9c-8682-4edfd9dd0143
```

Note: This ServerID can be used but it does present a problem with troubleshooting and making complex replication agreements. One thing that you can do is to change this number into a more defined name that you can used to better understand and create replication agreements. One main thing that you need to make sure is that these names *need to be unique*. For example:

```
#ibm-slapdServerId: 1320e26e-3457-4e9c-8682-4edfd9dd0143
ibm-slapdServerId: win2k1uid
```

The best time to do this is when you first install the directory and *before* you config the directory with ldapxcfg. Edit the ibmslapd.conf file and add the following line after ibm-slapdPwEncryption: imask -

```
ibm-slapdServerId: win2k1uid
```

Then save the file and go back to ldapxcfg and finish the configuration of the directory.

13.2 Major replication topologies

The major replication topologies found in IBM Tivoli Directory server can be one of the following types. But these can be more or less complex depending upon the number or directory servers participating in the topology.

13.2.1 Simple master-replica topology

It is the simplest of all topologies. It consists of one Master (supplier) directly propagating the changes to a replica (consumer) server. In case of multiple masters updating a single replica server, each subtree in the replica should receive updates from only one master. Otherwise, changes made by one master on the replica will be overwritten by the other. Two masters writing to the same subtree in the replica is only possible in case of peer to peer topology.

In Figure 13-2, M1 is the master and R1 and R2 are two replicas.
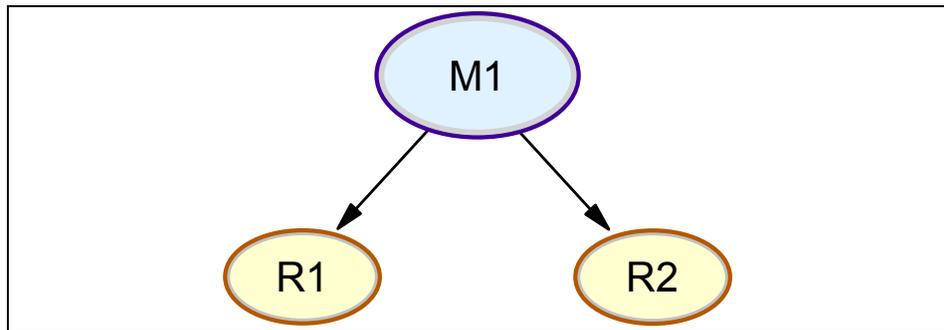
Figure 13-2   Simple master-replica scenario

13.2.2 Master-forwarder-replica topology (ITDS 5.2 and later)

In this scenario, the master (supplier) does not send the updates to the replica directly. Another server, called the Forwarder, lies between the Master and the replica and all changes from the master come via the forwarder to the replica. Hence, two replication agreements, one from the master to the forwarder and the other from the forwarder to the replica have to be created. The forwarder is read-only copy and cannot have peers. The forwarder itself maintains copy of the data that it receives from the master and then replicates it to the replicas. Such a replication scenario is also called Cascading Replication.

The advantages of using a forwarder is that the master need not replicate the changes to all the replicas. It sends the changes to the forwarder and its the responsibility of the forwarder to push them to all the replicas. Thus the master is free to do other tasks. Having a forwarder causes redundancy of data to deal

with any catastrophic incident resulting in loss of data. Also, the forwarder can be promoted to a master in case all data from the master is lost.

A forwarder is created by inserting a replica beneath an existing replica. The existing replica becomes a forwarder.

In Figure 13-3, M1 is the master, F1 was the previous replica, which is now a forwarder and R1 is the new replica which was inserted below the existing replica to convert it into a forwarder.
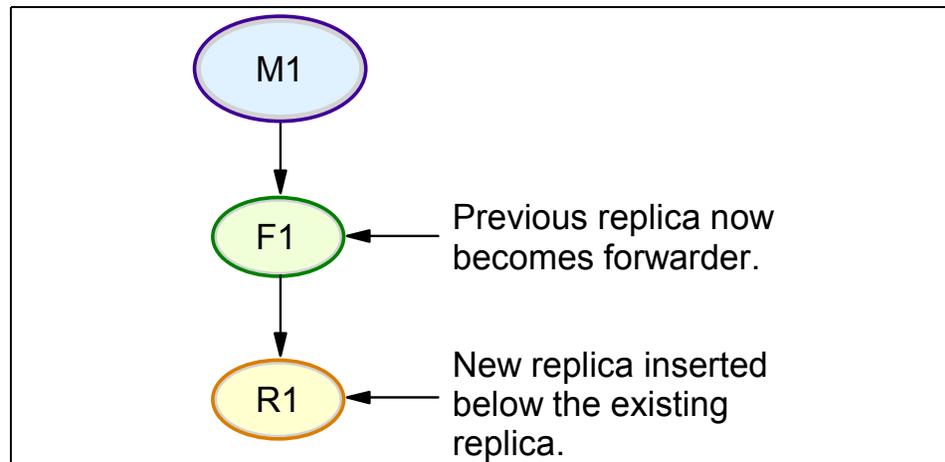
Figure 13-3   Master-forwarder replica topology

The main advantages of using peer servers in replication is performance enhancement (by providing local servers for accepting updates in widely distributed network) and reliability (since a peer server acts as a backup server for the other peer server).

13.2.3 GateWay Replication Topology (ITDS 5.2 and later)

Gateway servers are special types of peer servers which replicate changes received only from remote replication sites to all masters/replicas in the local replication site. A replication site is collection of masters/peers, gateways, forwarders and replicas. A master/peer is converted into a gateway server by adding the *ibm-replicaGateway* auxiliary class to the ibm-replicaSubentry entry corresponding to the master/peer. A replica or forwarder cannot be a gateway server.

The main advantage of using gateway servers is to reduce network traffic between two replication sites. If the replication sites are located in different LANs, the network traffic between the LANs is reduced. Gateway servers replicate the

changes to all the servers, including peers, in the local site but not to other gateway servers.

Figure 13-4 shows an example topology using gateway servers.
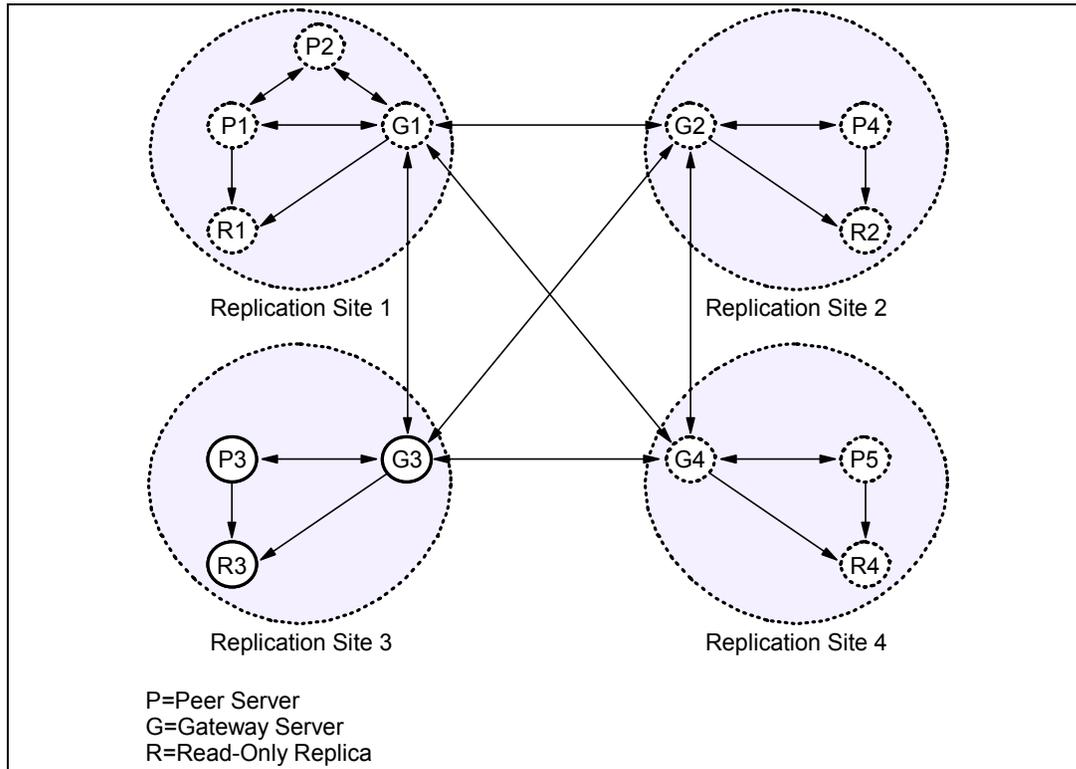
Figure 13-4   Replication topology with gateway servers

13.2.4 Peer replication

With any project design that would consist of two or more Peer to Peer servers for fault tolerant 24x7x365 uptime. All directory operations will be performed on one Peer server. These changes will be replicated automatically and immediately to the second or more peers. If the first peer fails, or needs to be taken offline for maintenance, the application can be reconfigure to route directory protocol traffic to the another peer. The other peers will queue any updates they process and forward them to the first peer as soon as it is brought back online.

Peer replication is an environment in which more that one LDAP server accepts updates from clients. The servers accept changes from other peer servers and

apply them to their own copies of the DIT. The changes are applied in the order they receive them. A peer to peer LDAP configuration can often be the most effective way to deploy the IBM Tivoli Directory Server LDAP to meet the business requirement for an enterprise directory service to be continually available for update on a 24x7 basis. A Multiple Peer configuration avoids loss of directory updates capability due to the failure of a single master server, hardware failure or site disaster, or the need to take the master LDAP server offline for service.

Update conflict prevention in peer configurations

Peer replication between IBM Directory Server peers may be used when the access patterns of the applications that update directory entries are controlled to prevent update conflicts. The LDAP name space and applications must be designed to ensure that the same directory entries will not be updated concurrently by clients on different peers. This is normally done by using a load balancer type device that can be either hardware or software controlled.

How peer to peer works

When a peer receives an update from a client, it makes the change to its own DIT and forwards the change to all the other Peer and Replica servers it knows. A peer must be configured to know about all other peers and replica servers that should be updated when it accepts an update from a client and how to connect to them.

Updates that are received from peer servers are not propagated to any other replica or peer server definitions. When an update is received by a peer server, the update is applied to the database. If the update was received from a peer server, the update is applied and processing stops. If the update was made by another client, the directory is updated, and the update information is propagated to the other peers and replicas. This prevents peers from continually replicating an update to each other. Because peer servers do not propagate updates to other replica definitions.

A peer maintains a copy of each change it accepts from a client in its replication queue until it receives a positive acknowledgement from all other peers and replicas that the change has been accepted. If a peer cannot contact another peer or replica, it will log the problem in its slapd.errors file and keep the change in its replication queue. It will periodically attempt to connect to the failed server until the problem is corrected. Therefore, a failed peer server should be unconfigured if it is not going to be restarted.

At startup, a server queries the database to determine if there are any replica objects. These objects define replica or peer servers for this particular server.

The schema definition for these objects describes how a server can locate and connect to the replica or peer server, as well as other replication properties.

Configuration for peer to peer in IBM Directory 4.1 and earlier

Within the slapd32.conf file there are currently several parameters relating to replication. If an `ibm-slapdMasterServerDn` and `ibm-slapdMasterServerPW` are specified, this server is presumed to be a read-only replica. The `ibm-slapdMasterServerDn` and `ibm-slapdMasterServerPW` in the slapd32.conf file must match the `replicaBindDn` and `replicaCredentials` in the replica object definition found on the master server. A peer server is designated in the slapd32.conf file with the attributes `ibm-slapdPeerDn` and ibm-slapdPeerPW in the `cn=Master,cn=Configuration` object.

> **Note**: The ibm-slapdPeerDn cannot be the same as the ibm-slapdAdminDn for peer replication to function correctly. This designates a writable copy of the database.

Add a replica definition to all servers within the peer network representing each of the other peer and replica servers in the network. This is different from the replica, which does not have any replica definitions within the database. The ibm-slapdPeerDn and the ibm-slapdPeerPW in a server's slapd32.conf file must match the replicaBindDn and replicaCredentials in replica objects on the other peer servers that bind to it. Because these passwords must match, the same peer replica object definitions can be used on all peer servers.

A server can be either a replica server or a peer server, it can not be both. That means that the ibm-slapdMasterServerDn parameter in the slapd32.conf file is mutually exclusive with the ibm-slapdPeerDn configuration file attribute. If both are defined in the slapd32.conf file, the server does not start and the following message is logged in the slapd.errors file: Can not specify both masterServerDn and peerDn

Configuration for Multiple Peer to Peer from a normal master/slave as shown in the next two figures. The first figure (Figure 13-5 on page 329) shows how the master/slave was set up the next figure (Figure 13-6 on page 330) shows how peer to peer will look like when it is configured.
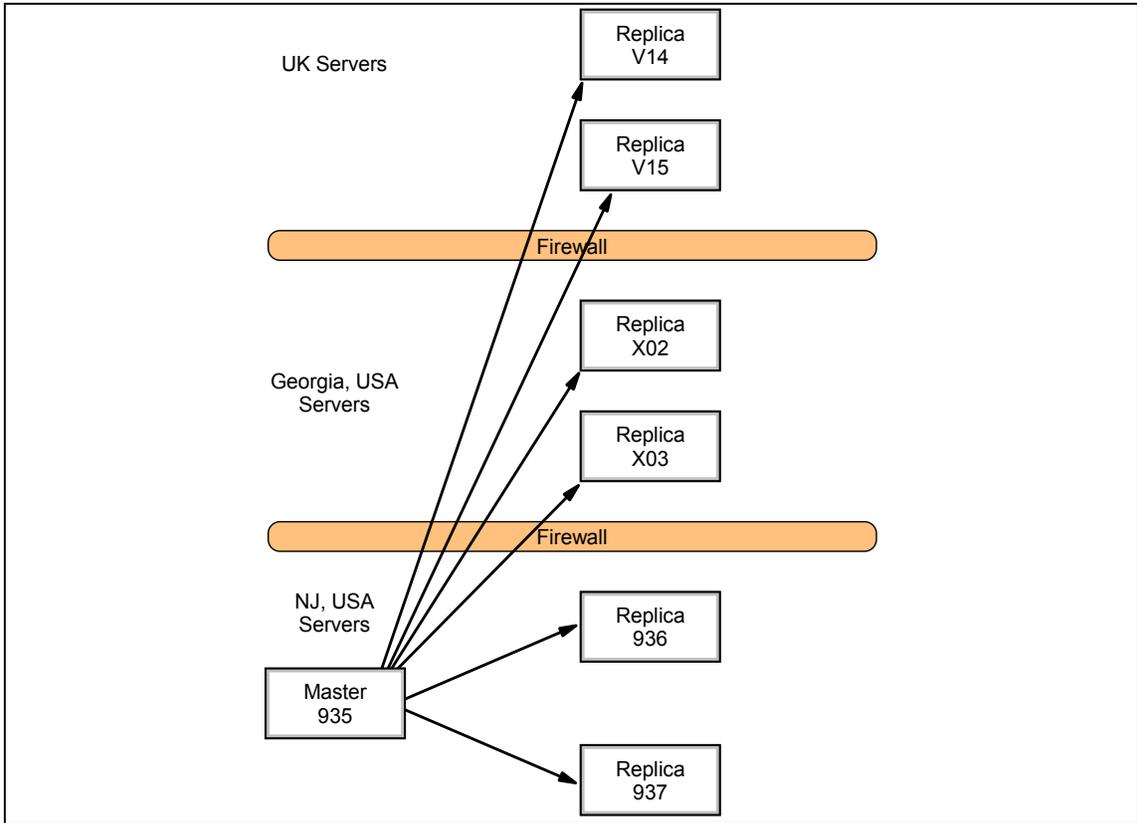
Figure 13-5   Original LDAP flow
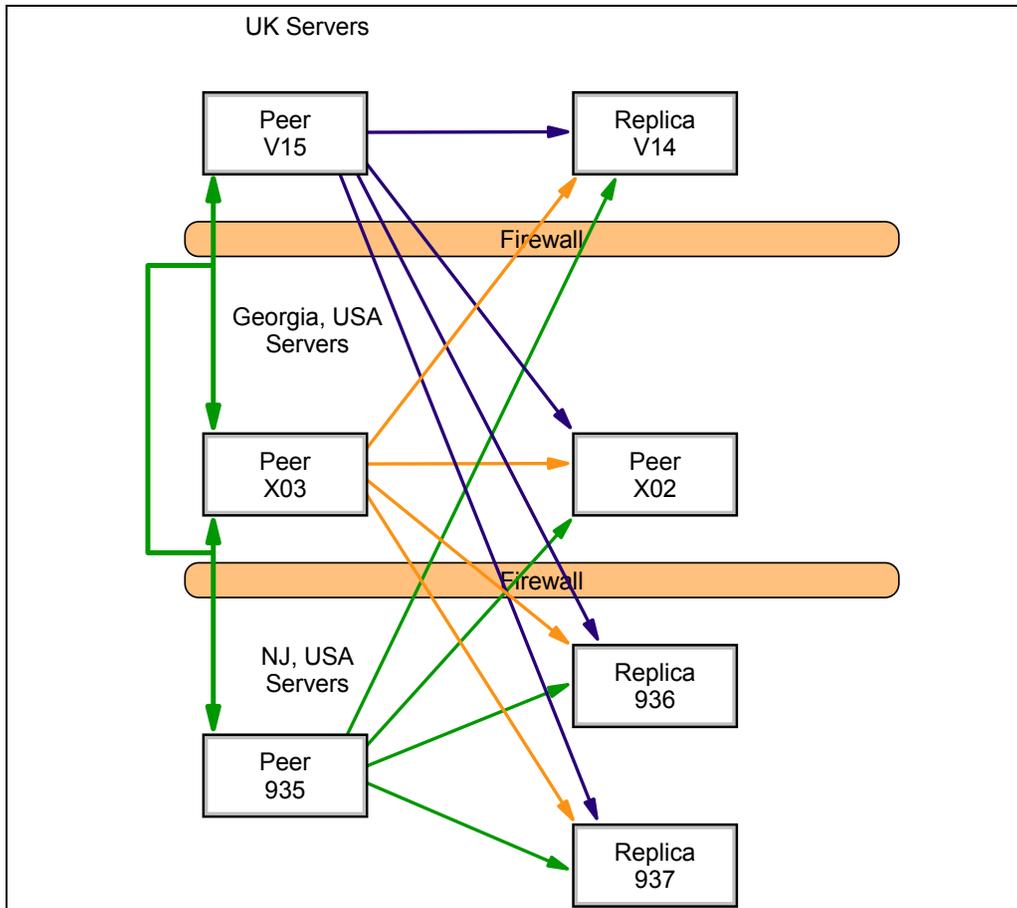
Figure 13-6   Multiple peer LDAP flow

Peer 935 listens on port 636 (using SSL), and is a peer server of Peer X03 and Peer V15 in (Figure 13-6). Before we put in Replica X03 and Replica V15 as Peer servers (Figure 13-5 on page 329), you will have an entry dn: cn=Master Server, cn=Configuration in the slapd32.conf file. This tells us that these servers are Replicas. What will be needed is to turn off replication, and stop the server before proceeding.

Peer 1 is configured to recognize Peer 2 and 3 as a peer and the password Peer 2 and 3 will use to bind to Peer 1 by the following LDIF file. This LDIF file updates the slapd32.conf file. This configuration may also be added to the configuration file manually. If you do update the file manually, you should always make a backup copy of the file first. All three machine peers must use the same ID and passwords.

```
dn:cn=Master Server, cn=Configuration
cn:Master Server
ibm-slapdPeerDn:cn=peer
ibm-slapdPeerPW:< same peer password>
objectclass:ibm-slapdReplication
objectclass:top
```

A replica object is added to the Peer 1 database through the following ldif file. The replica object is added to the cn=localhost suffix of the database. It specifies the bind DN and password that Peer 1 will use when it binds as a peer to Peer 2 and 3. You would also add all the replica info of the other replicas in your tree to this list (not shown).

```
dn: cn=Peer2, cn=localhost
cn: Peer2
replicaBindDN: cn=peer
replicaCredentials: <same peer password>
replicaPort: 636
replicaHost: <fully-qualified-hostname>
replicaBindMethod: Simple
replicaUseSSL: TRUE
replicaUpdateTimeInterval: 0
objectclass: replicaObject
objectclass: top

dn: cn=Peer3, cn=localhost
cn: Peer3
replicaBindDN: cn=peer
replicaCredentials: <same peer password>
replicaPort: 636
replicaHost: <fully-qualified-hostname>
replicaBindMethod: Simple
replicaUseSSL: TRUE
replicaUpdateTimeInterval: 0
objectclass: replicaObject
objectclass: top
```

If you are not using SSL for your LDAP do the following:

```
dn: cn=Peer2, cn=localhost
cn: Peer2
replicaBindDN: cn=peer
replicaCredentials: <same peerpassword>
replicaPort: 389
replicaHost: <fully-qualified-hostname>
replicaBindMethod: Simple
replicaUseSSL: FALSE
replicaUpdateTimeInterval: 0
objectclass: replicaObject
objectclass: top
```

```
dn: cn=Peer3, cn=localhost
cn: Peer3
replicaBindDN: cn=peer
replicaCredentials: <same peerpassword>
replicaPort: 389
replicaHost: <fully-qualified-hostname>
replicaBindMethod: Simple
replicaUseSSL: FALSE
replicaUpdateTimeInterval: 0
objectclass: replicaObject
objectclass: top
```

Peer 2 and 3 listens on port 636 (using SSL), and is a peer server of Peer 1. This
configuration may also be added to the configuration file manually. If you do
update the file manually, you should always make a backup copy of the file first.
All three machine peers must use the same ID and passwords.

```
dn:cn=Master Server, cn=Configuration
cn:Master Server
ibm-slapdPeerDn:cn=peer
ibm-slapdPeerPW:< same peer password>
objectclass:ibm-slapdReplication
objectclass:top
```

A replica object is added to the Peer 2 database through the following ldif files. It
specifies the bind DN and password that Peer 2 will use to bind as a peer to Peer
1. This must match the information in the slapd32.conf file for Peer 1. You would
also add all the replica info of the other replicas in your tree to this list (not
shown).

```
dn: cn=Peer1, cn=localhost
cn: Peer1
replicaBindDN: cn=peer
replicaCredentials: <same peer password>
replicaPort: 636
replicaHost: <fully-qualified-hostname>
replicaBindMethod: Simple
replicaUseSSL: TRUE
replicaUpdateTimeInterval: 0
objectclass: replicaObject
objectclass: top

dn: cn=Peer3, cn=localhost
cn: Peer3
replicaBindDN: cn=peer
replicaCredentials: <same peer password>
replicaPort: 636
replicaHost: <fully-qualified-hostname>
```

```
replicaBindMethod: Simple
replicaUseSSL: TRUE
replicaUpdateTimeInterval: 0
objectclass: replicaObject
objectclass: top
```

If you are not using SSL for your LDAP do the following:

```
dn: cn=Peer1, cn=localhost
cn: Peer1
replicaBindDN: cn=peer
replicaCredentials: <same peer password>
replicaPort: 389
replicaHost: <fully-qualified-hostname>
replicaBindMethod: Simple
replicaUseSSL: FALSE
replicaUpdateTimeInterval: 0
objectclass: replicaObject
objectclass: top

dn: cn=Peer3, cn=localhost
cn: Peer3
replicaBindDN: cn=peer
replicaCredentials: <same peer password>
replicaPort: 389
replicaHost: <fully-qualified-hostname>
replicaBindMethod: Simple
replicaUseSSL: FALSE
replicaUpdateTimeInterval: 0
objectclass: replicaObject
objectclass: top
```

A replica object is added to the Peer 3 database through the following ldif files. It specifies the bind DN and password that Peer 3 will use to bind as a peer to Peer 1. This must match the information in the slapd32.conf file for Peer 1. You would also add all the replica info of the other replicas in your tree to this list (not shown).

```
dn: cn=Peer1, cn=localhost
cn: Peer1
replicaBindDN: cn=peer
replicaCredentials: <same peer password>
replicaPort: 636
replicaHost: <fully-qualified-hostname>
replicaBindMethod: Simple
replicaUseSSL: TRUE
replicaUpdateTimeInterval: 0
objectclass: replicaObject
objectclass: top
```

```
dn: cn=Peer2, cn=localhost
cn: Peer2
replicaBindDN: cn=peer
replicaCredentials: <same peer password>
replicaPort: 636
replicaHost: <fully-qualified-hostname>
replicaBindMethod: Simple
replicaUseSSL: TRUE
replicaUpdateTimeInterval: 0
objectclass: replicaObject
objectclass: top
```

If you are not using SSL for your LDAP do the following:

```
dn: cn=Peer1, cn=localhost
cn: Peer1
replicaBindDN: cn=peer
replicaCredentials: <same peer password>
replicaPort: 389
replicaHost: <fully-qualified-hostname>
replicaBindMethod: Simple
replicaUseSSL: FALSE
replicaUpdateTimeInterval: 0
objectclass: replicaObject
objectclass: top

dn: cn=Peer2, cn=localhost
cn: Peer2
replicaBindDN: cn=peer
replicaCredentials: <same peer password>
replicaPort: 389
replicaHost: <fully-qualified-hostname>
replicaBindMethod: Simple
replicaUseSSL: FALSE
replicaUpdateTimeInterval: 0
objectclass: replicaObject
objectclass: top
```

Change replicas and original master server into Peer Servers

First you will have to make sure that the replica is sync up with the master and
there are no changes pending.

On the original master server you will need to put it into read-only mode. And
restart the server.   Then break the replication agreement with the replica that
you are going to change into a peer server. For our example this would be UK
Replica V15 and Georgia X03 servers.

Create three ldif's to configure the Peer Servers, one for Peer1 and one for Peer2 and one for Peer3. See the following LDIF examples in this document (on page 336). These LDIF examples will include the replica information of the other 4 replicas.

Shut down the slapd process on the original master server NJ 935.

Use **ldif2db** command to load the following ldif file. Using the ldif2db with the slapd process shutdown will input the replication peer data into the servers when they are down. This way when the slapd process is brought back up it will know of the other servers and set them up in the DB2 database and start saving any changes in the change tables.

```
ldif2db -i  e:\migration\peer1.ldif
```

Make needed changes to the slapd32.conf file.

```
dn:cn=Master Server, cn=Configuration
cn:Master Server
ibm-slapdPeerDn:cn=peer
ibm-slapdPeerPW:< same peer password>
objectclass:ibm-slapdReplication
objectclass:top
```

Restart slapd process on the new Peer 935 server.

Shut down the slapd process on the Georgia X03 server.

Use the **ldif2db** command to load the following ldif file.

```
ldif2db -i  e:\migration\peer2.ldif
```

Make the needed changes to the slapd32.conf file for the Georgia X03 server.

```
dn:cn=Master Server, cn=Configuration
cn:Master Server
ibm-slapdPeerDn:cn=peer
ibm-slapdPeerPW:< same peer password>
objectclass:ibm-slapdReplication
objectclass:top
```

Restart slapd process on the new Peer X03 server.

Shut down the slapd process on the UK V15 server.

Use the **ldif2db** command to load the following ldif file.

```
ldif2db -i  e:\migration\peer3.ldif
```

Make the needed changes to the slapd32.conf file for the UK V15 server.

```
dn:cn=Master Server, cn=Configuration
cn:Master Server
ibm-slapdPeerDn:cn=peer
ibm-slapdPeerPW:< same peer password>
objectclass:ibm-slapdReplication
objectclass:top
```

Restart the slapd process on the new Peer V15 server.

Check the slapd.errors file on the Peer 935 server to make sure that it has connected back up to all the six servers.

Make a change on one of the peers and then check to see that the change went to the other peers.

Then do it in reverse from each of the other two peers make a different change to make sure changes are made all three ways.

Reconfigure the remaining replicas in the UK and Georgia to now refer there write traffic to the new peer server in each of there respective sites. UK replica to the UK peer and the Georgia replica to the Georgia peer.

Now a test needs to be made to check to make sure that the other sites will work with out the NJ site. This will be done by bringing down the Peer 935 slapd process and the Application process and then trying to log into the UK and the Georgia sites and authenticating to there applications. This is to simulate the main site (NJ) going down due to power outages.

Peer LDIF files
The following shows the peer LDIF files for NJ, Georgia, and the UK sites:

► NJ Peer 935 Peer1.ldif
```
dn: cn=xxgasrv03, cn=localhost
cn: xxgasrv03
replicaBindDN: cn=ibmpeer
replicaCredentials: XXXXXXX
replicaPort: 636
replicaHost: xxgasrv03.us.ibm.com
replicaBindMethod: Simple
replicaUseSSL: TRUE
replicaUpdateTimeInterval: 0
objectclass: replicaObject
objectclass: top

dn: cn=gouksrv15, cn=localhost
cn: gouksrv15
```

```
replicaBindDN: cn=ibmpeer
replicaCredentials: XXXXXXX
replicaPort: 636
replicaHost: gouksrv15.uk.ibm.com
replicaBindMethod: Simple
replicaUseSSL: TRUE
replicaUpdateTimeInterval: 0
objectclass: replicaObject
objectclass: top

dn: cn=usnj936,cn=localhost
cn: usnj936
replicahost: usnj936.us.ibm.com
replicabinddn: cn=usuk936
replicacredentials: XXXXXXX
replicaport: 636
replicabindmethod: SIMPLE
replicausessl: TRUE
replicaupdatetimeinterval: 0
seealso::
description::
objectclass: replicaObject
objectclass: top

dn: cn=usnj937,cn=localhost
cn: usnj937
replicahost: usnj937.us.ibm.com
replicabinddn: cn=usnj937
replicacredentials: XXXXXXX
replicaport: 636
replicabindmethod: SIMPLE
replicausessl: TRUE
replicaupdatetimeinterval: 0
seealso::
description::
objectclass: replicaObject
objectclass: top

dn: cn=xxgasrv02,cn=localhost
cn: xxgasrv02
replicahost: xxgasrv02.us.ibm.com
replicabinddn: cn=xxgasrv02
replicacredentials: XXXXXXX
replicaport: 636
replicabindmethod: SIMPLE
replicausessl: TRUE
replicaupdatetimeinterval: 0
seealso::
description::
```

```
objectclass: replicaObject
objectclass: top

dn: cn=gouksrv14,cn=localhost
cn: gouksrv14
replicahost: gouksrv14.uk.ibm.com
replicabinddn: cn=gouksrv14
replicacredentials: XXXXXXX
replicaport: 636
replicabindmethod: SIMPLE
replicausessl: TRUE
replicaupdatetimeinterval: 0
seealso::
description::
objectclass: replicaObject
objectclass: top
```

▶ Georgia USA Peer X03 Peer2.ldif

```
dn: cn=usnj935, cn=localhost
cn: usnj935
replicaBindDN: cn=ibmpeer
replicaCredentials: XXXXXXX
replicaPort: 636
replicaHost: usnj935.us.ibm.com
replicaBindMethod: Simple
replicaUseSSL: TRUE
replicaUpdateTimeInterval: 0
objectclass: replicaObject
objectclass: top

dn: cn=gouksrv15, cn=localhost
cn: gouksrv15
replicaBindDN: cn=ibmpeer
replicaCredentials: XXXXXXX
replicaPort: 636
replicaHost: gouksrv15.us.ibm.com
replicaBindMethod: Simple
replicaUseSSL: TRUE
replicaUpdateTimeInterval: 0
objectclass: replicaObject
objectclass: top

dn: cn=usnj936,cn=localhost
cn: usnj936
replicahost: usnj936.us.ibm.com
replicabinddn: cn=usnj936
replicacredentials: XXXXXXX
replicaport: 636
replicabindmethod: SIMPLE
```

```
replicausessl: TRUE
replicaupdatetimeinterval: 0
seealso::
description::
objectclass: replicaObject
objectclass: top

dn: cn=usnj937,cn=localhost
cn: usnj937
replicahost: usnj937.us.ibm.com
replicabinddn: cn=usnj937
replicacredentials: XXXXXXX
replicaport: 636
replicabindmethod: SIMPLE
replicausessl: TRUE
replicaupdatetimeinterval: 0
seealso::
description::
objectclass: replicaObject
objectclass: top

dn: cn=xxgasrv02,cn=localhost
cn: xxgasrv02
replicahost: xxgasrv02.us.ibm.com
replicabinddn: cn=xxgasrv02
replicacredentials: XXXXXXX
replicaport: 636
replicabindmethod: SIMPLE
replicausessl: TRUE
replicaupdatetimeinterval: 0
seealso::
description::
objectclass: replicaObject
objectclass: top

dn: cn=gouksrv14,cn=localhost
cn: gouksrv14
replicahost: gouksrv14.uk.ibm.com
replicabinddn: cn=gouksrv14
replicacredentials: XXXXXXX
replicaport: 636
replicabindmethod: SIMPLE
replicausessl: TRUE
replicaupdatetimeinterval: 0
seealso::
description::
objectclass: replicaObject
objectclass: top
```

▶ UK Peer V15 Peer3.ldif

```
dn: cn=xxgasrv03, cn=localhost
cn: xxgasrv03
replicaBindDN: cn=ibmpeer
replicaCredentials: XXXXXXX
replicaPort: 636
replicaHost: xxgasrv03.us.ibm.com
replicaBindMethod: Simple
replicaUseSSL: TRUE
replicaUpdateTimeInterval: 0
objectclass: replicaObject
objectclass: top

dn: cn=usnj935, cn=localhost
cn: usnj935
replicaBindDN: cn=ibmpeer
replicaCredentials: XXXXXXX
replicaPort: 636
replicaHost: usnj935.us.ibm.com
replicaBindMethod: Simple
replicaUseSSL: TRUE
replicaUpdateTimeInterval: 0
objectclass: replicaObject
objectclass: top

dn: cn=usnj936,cn=localhost
cn: usnj936
replicahost: usnj936.us.ibm.com
replicabinddn: cn=usnj936
replicacredentials: XXXXXXX
replicaport: 636
replicabindmethod: SIMPLE
replicausessl: TRUE
replicaupdatetimeinterval: 0
seealso::
description::
objectclass: replicaObject
objectclass: top

dn: cn=usnj937,cn=localhost
cn: usnj937
replicahost: usnj937.us.ibm.com
replicabinddn: cn=usnj937
replicacredentials: XXXXXXX
replicaport: 636
replicabindmethod: SIMPLE
replicausessl: TRUE
replicaupdatetimeinterval: 0
```

```
        seealso::
        description::
        objectclass: replicaObject
        objectclass: top

        dn: cn=xxgasrv02,cn=localhost
        cn: xxgasrv02
        replicahost: xxgasrv02.us.ibm.com
        replicabinddn: cn=xxgasrv02
        replicacredentials: XXXXXXX
        replicaport: 636
        replicabindmethod: SIMPLE
        replicausessl: TRUE
        replicaupdatetimeinterval: 0
        seealso::
        description::
        objectclass: replicaObject
        objectclass: top

        dn: cn=gouksrv14,cn=localhost
        cn: gouksrv14
        replicahost: gouksrv14.uk.ibm.com
        replicabinddn: cn=gouksrv14
        replicacredentials: XXXXXXX
        replicaport: 636
        replicabindmethod: SIMPLE
        replicausessl: TRUE
        replicaupdatetimeinterval: 0
        seealso::
        description::
        objectclass: replicaObject
        objectclass: top
```

Peer-to-peer replication topology for ITDS 5.1 and later

Peer servers are Masters which not only propagate changes to replicas and
forwarders below them but also receive changes from other master servers.
Hence, peers are read-write replicas. Starting with ITDS 5.1, peers are
configured in exactly the same way as the master servers and the terms Peer
and Master can be used interchangeably.

Peer servers replicate all client updates but do not replicate updates received
from other masters/peers. Client update refers to updates made by a bind DN
other than the Master ServerDN (represented by ibm-slapdMasterDN attribute in
the config file).

An example peer-to-peer replication topology is shown in Figure 13-7 on
page 342.
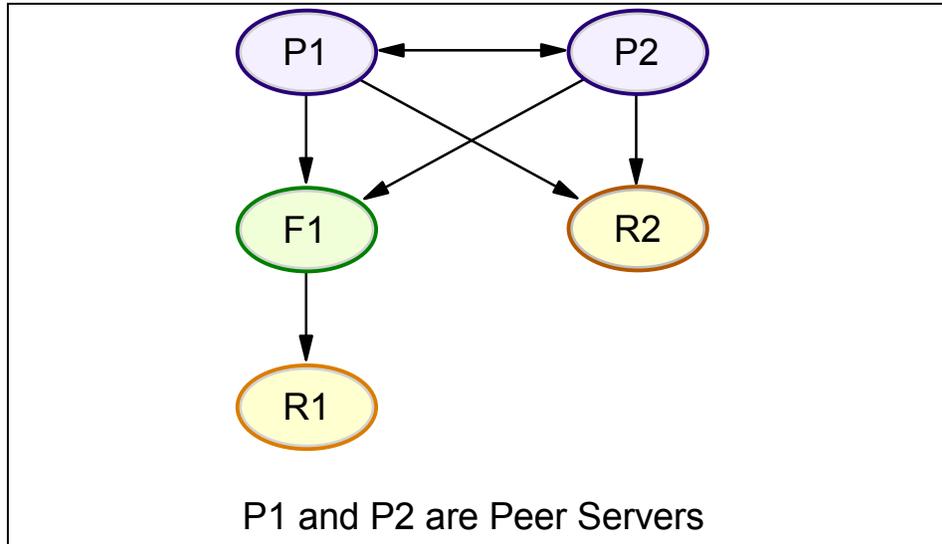
Figure 13-7   Peer-to-peer replication topology

13.3 Replication agreements

A replication agreement is an entry in the directory with the object class ibm-replicationAgreement created beneath a replica subentry to define replication from the server represented by the subentry to another server. These objects are similar to the replicaObject entries used by ITDS 4.1 and earlier.

The replication agreement consists of the following items:

- ► A user friendly name, used as the naming attribute for the agreement. This name might be the consumer server name or some other descriptive string.

- ► An LDAP URL specifying the server, port number, and whether SSL should be used.

- ► The consumer server id, if ITDS 5.1 and later will be defined in the ibmslapd.conf file as the ibm-slapdServerId, It will show *unknown* for a server whose server ID is not known as in a server running on IDS 4.1 and earlier. The consumer server id is used by the administrative GUI to traverse the topology. Given the consumer's server ID, GUI can find the corresponding subentry and its agreements.

- ► The DN of an object containing the credentials used by the supplier to bind to the consumer. Because the replication agreement can be replicated, a DN to a credentials object is used. This allows the credentials to be stored in a nonreplicated area of the directory, like the cn=localhost. Replicating the

credentials objects (from which *clear text* credentials must be obtainable) represents a potential security exposure. Use of a separate object also makes it easier to support various authentication methods; new object classes can be created rather than trying to make sense of numerous optional attributes.

► An optional DN pointer to an object containing the schedule information for replication. If the attribute is not present, changes are replicated immediately.

► You can designate that part of a replicated subtree not be replicated by adding the ibm-replicationContext auxiliary class to the root of the subtree, without defining any replica subentries.

> **Note:** The Web Administration Tool also refers to agreements as *queues* when referring to the set of changes that are waiting to be replicated under a given agreement.

13.4 Configuring replication topologies

The following section describes the steps required for configuring IBM Tivoli Directory 5.2 server with the following examples:

1. One master with two replicas. And the directory has two suffixes.

2. One main master with two peer servers for one suffix and two peer servers for another suffix.

3. Sub tree replication where you take a non-suffix container and have a master with one replica for that sub tree.

For more configuring other types of topologies please refer to the IBM Tivoli Directory Server 5.2 administration guide at:

http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html

13.4.1 Simple master-replica topology

Configuring a simple master-replica scenario involves the following four steps:

1. Choose one server to act as the master and select the subtree in it to be replicated. For our example we will be having one master and two replicas with each replica being a replica on one of the two suffixes.

2. Create credentials to be used by the Master server.

3. Create replica servers.

4. Export data to the replica servers.

Using the Web Administration Tool

Note: If you are trying to make a non-suffix entry the replicated root, for example a sub container that is under the suffix, the following steps need to be done before the Add Subtree function is used.

Go to the Manage Entries panel. Select the entry and click **Edit ACL**. If you want to add Non-filtered ACLs, select that tab and add an entry cn=this with the role access-id for both ACLs and owners. Ensure that Propagate ACLs and Propagate owner are checked. If you want to add Filtered ACLs select that tab and add an entry cn=this with the role access-id for both ACLs and owners. Ensure that Accumulate filtered ACLs is unchecked and that Propagate owner is checked.

For manual loading by way of a ldif. you will need to add the following to the DN that you want to replicate.

For non-filtered ACLs:

```
ownersource: <same as the entry DN>
ownerpropagate: TRUE

aclsource: <same as the entry DN>
aclpropagate: TRUE
```

For filtered ACLs you will need to add the following:

```
ibm-filteraclinherit: FALSE
```

The above steps are not required for a suffix entry since a suffix gets all these ACLs by default.

Creating the Master Server

This task designates an entry as the root of an independently replicated subtree and creates a ibm-replicasubentry representing this server as the single master for the subtree. To create a replicated subtree, you must designate the subtree that you want the server to replicate.

Note: On the Linux, Solaris, and HP-UX platforms, if a referral fails because the server being referred to is not running, ensure that the environment variable LDAP_LOCK_REC has been set in your system environment. No specific value is required.

```
set LDAP_LOCK_REC=anyvalue
```
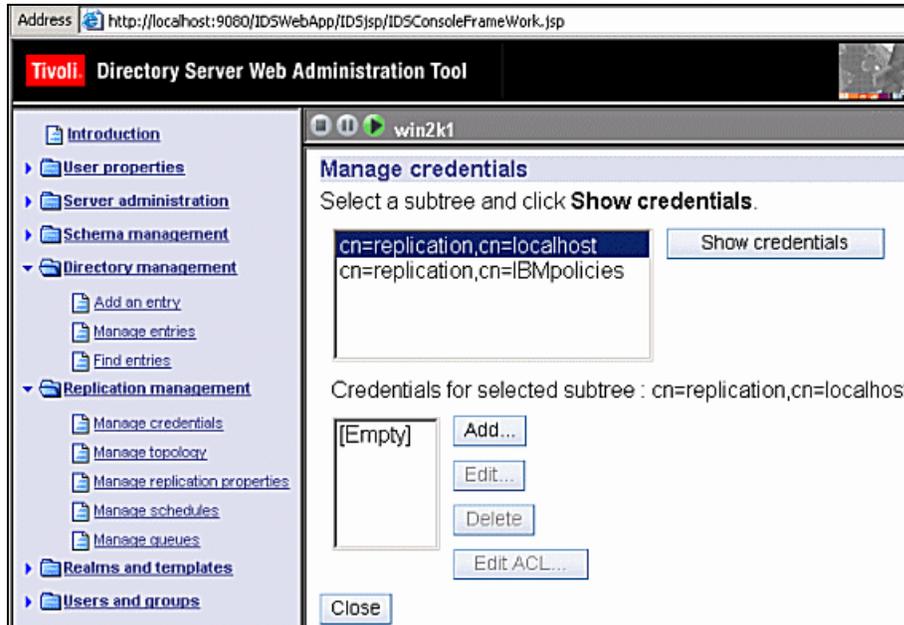
Figure 13-8   Web Admin Tool - Manage credentials

Creating credentials

Expand the Replication management category in the navigation area of the Web Administration Tool and click **Manage credentials**.

1. Select the location that you want to use to store the credentials from the list of subtrees. The Web Administration Tool allows you to define credentials in three locations:

    –   cn=replication,cn=localhost, which keeps the credentials only on the current server.

> **Note:** In most replication cases, locating credentials in
> cn=replication,cn=localhost is preferred because it provides greater
> security than replicated credentials located on the subtree. If you are
> going to do this you will need to export the cn=replication,cn=localhost
> like the following:
>
> ```
> cn=replication,cn=localhost
> objectclass=container
> objectclass=top
> cn=replication
>
> cn=masterbind,cn=replication,cn=localhost
> replicaCredentials=secret01
> description=master bind credential
> objectclass=ibm-replicationCredentials
> objectclass=ibm-replicationCredentialsSimple
> objectclass=top
> replicaBindDN=cn=masterbind
> cn=masterbind
> ```
>
> To each of the other replicas so they will have the credentials.n
> situations in which credentials located on cn=replication,cn=localhost
> are not available.

If you are trying to add a replica under a server, for example server A and
you are connected to a different server with the Web Administration Tool,
server B, the Select credentials field does not display the option
cn=replication,cn=localhost. This is because you cannot read the
information or update any information under cn=localhost of the server A
when you are connected to server B.The cn=replication,cn=localhost is
only available when the server under which you are trying to add a replica
is the same server that you are connected to with the Web Administration
Tool.

– cn=replication,cn=IBMpolicies, which is available even when the server
under which you are trying to add a replica is not the same server that you
are connected to with the Web Administration Tool. Credentials placed
under this location are replicated to the servers.

> **Note:** The location cn=replication,cn=IBMpolicies is only available, if
> the IBMpolicies support OID, 1.3.18.0.2.32.18, is present under the
> ibm-supportedcapabilities of the root DSE.

- Within the replicated subtree, in which case the credentials are replicated with the rest of the subtree. Credentials placed in the replicated subtree are created beneath the ibm-replicagroup=default entry for that subtree.

  If no subtrees are displayed, go to "Creating the Master Server" on page 344 (replicated subtree) for instructions about creating the subtree that you want to replicate.
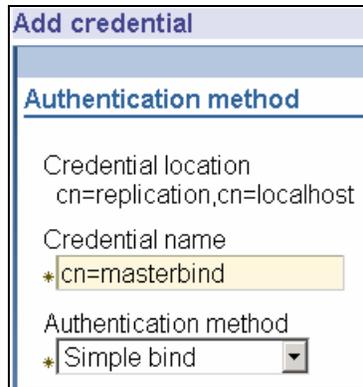
2. Click **Add**.

Figure 13-9   Add credential

3. Enter the name for the credentials you are creating, for example, `masterbind`, cn= is prefilled in the field for you, as shown in Figure 13-9.
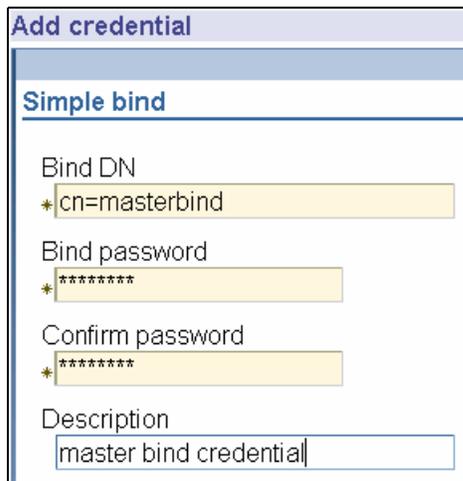
Figure 13-10   Simple bind

4. Select the type of authentication method you want to use and click **Next**, as shown in Figure 13-10 on page 347.

   – If you selected simple bind authentication:

   - Enter the DN that the server uses to bind to the replica, for example, cn=masterbind.

   - Enter the password uses when it binds to the replica, for example, secret.

   - Enter the password again to confirm that there are no typographical errors.

   - If you want, enter a brief description of the credentials.

   - Click **Finish**.

   – If you selected Kerberos authentication:

   - Enter your Kerberos bind DN.

   - Enter the bind password.

   - Reenter the bind password to confirm it.

   - If you want, enter a brief description of the credentials. No other information is necessary.

   - Click **Finish**.

     By default, the supplier uses its own service principal to bind with the consumer. For example, if the supplier is named master.our.org.com and the realm is SOME.REALM, the DN is ibm-Kn=ldap/master.our.org.com@SOME.REALM. The realm value is case insensitive. If there is more than one supplier, you must specify the principal and password to be used by all of the suppliers.

     On the server where you created the credentials:

     Expand **Directory management** and click **Manage entries**.

     Select the subtree where you stored the credentials, for example cn=localhost, and click **Expand**.

     Select **cn=replication** and click **Expand**.

     Select the kerberos credentials (ibm-replicationCredentialsKerberos) and click **Edit attributes**.

     Click the **Other attributes** tab.

     Enter the replicaBindDN, for example, ibm-kn=myprincipal@SOME.REALM.

Enter the replicaCredentials. This is the KDC password used for myprincipal. This principal and password should be the same as the ones you use to run `kinit` from the command line.

On Replica:

Click **Manage replication properties** in the navigation area.

Select a supplier from the Supplier information drop-down menu or enter the name of the replicated subtree for which you want to configure supplier credentials.

Click **Edit**.

Enter the replication bindDN. In this example, `ibm-kn=myprincipal@SOME.REALM`.

Enter and confirm the Replication bind password. This is the KDC password used for myprincipal**.**

If you selected SSL with certificate authentication you do not need to provide any additional information, if you are using the server's certificate. If you choose to use a certificate other than the server's:

Enter the key file name.

Enter the key file password.

Reenter the key file password to confirm it.

Enter the key label.

If you want, enter a brief description.

Click **Finish**.

5. Expand the **Replication Management** category in the navigation area and click **Manage topology**.
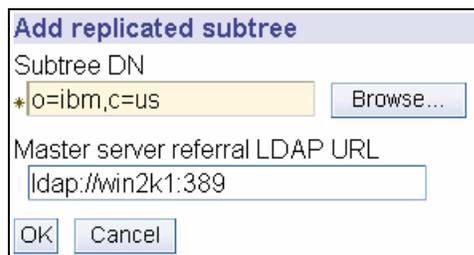
Figure 13-11   Add replicated subtree

a. Click **Add subtree** (the window in Figure 13-11 is shown).

b. Enter the DN of the subtree that you want to replicate or click Browse to expand the entries to select the entry that is to be the root of the subtree.

c. The master server referral URL is displayed in the form of an LDAP URL, for example ldap://<myservername>.<mylocation>.<mycompany>.com. This is optional and is used only if server contains (or will contain) any read-only subtrees.

d. To define a referral URL that is returned for updates to any read-only subtree on the server.

e. You could also just use the servername if you have defined in the hosts file or are using a DNS.

f. Click **OK**.

6. The new server is displayed on the Manage topology panel under the heading Replicated subtrees (Figure 13-12).
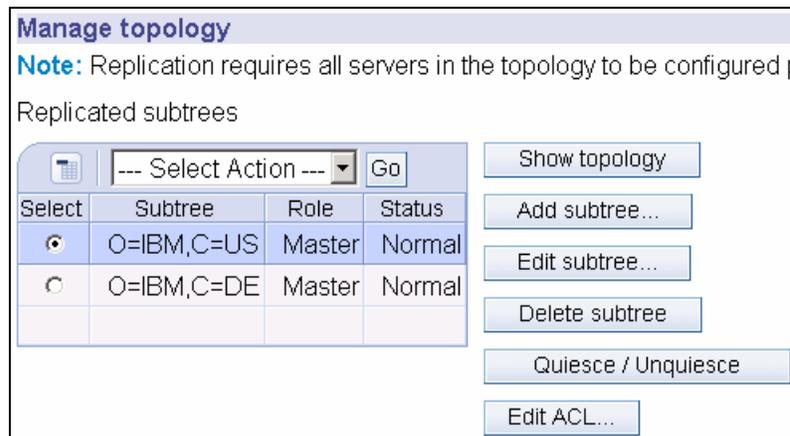
Figure 13-12   Manage topology

7. Create the Replica Server.
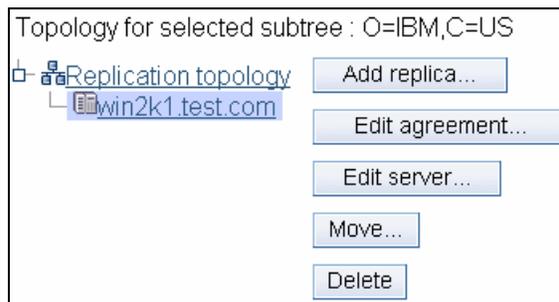
Figure 13-13   Show topology

Expand the **Replication management** category in the navigation area and click **Manage topology**, as shown in Figure 13-13 on page 350.

a. Select the subtree that you want to replicate and click **Show topology**.

b. Click the arrow next to the Replication topology selection to expand the list of supplier servers.

c. Select the supplier server and click **Add replica**.

Figure 13-14   Add replica

d. On the Server tab of the Add replica window (shown in Figure 13-14):

i. Enter the host name and port number for the replica you are creating. The default port is 389 for non-SSL and 636 for SSL. These are required fields.

ii. Select whether to enable SSL communications.

iii. Enter the replica name or leave this field blank to use the host name.

iv. Enter the replica ID. If the server on which you are creating the replica is running, click **Get replica ID** to automatically fill this field. This is a required field, if the server you are adding is going to be a peer or forwarding server. It is recommended for all IBM Tivoli Directory Server Version 5.2 replica servers.

v. Enter a description of the replica server.

vi. If a credential object is not selected in the Additional Tab, an error
message will be displayed as shown in Figure 13-15.
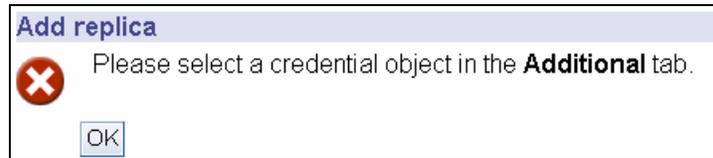
Figure 13-15   Error message when Additional is not used
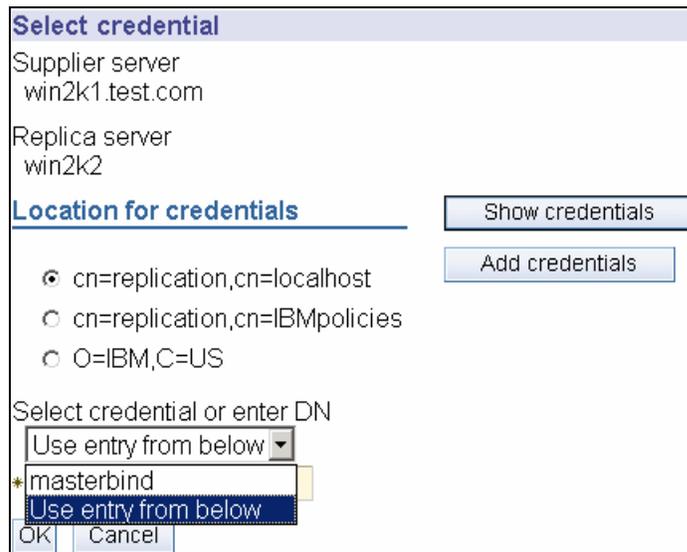
Figure 13-16   Additional tab - Select credential

8. On the Additional tab**,** shown in Figure 13-16**,** specify the credentials that the
replica uses to communicate with the master. The Web Administration Tool
allows you to define credentials in three places:

a. cn=replication,cn=localhost, which keeps the credentials only on the
server that uses them.

b. cn=replication,cn=IBMpolicies, which is available even when the server
under which you are trying to add a replica is not the same server that you
are connected to with the Web Administration Tool. Credentials placed
under this location are replicate to the servers. The location
cn=replication,cn=IBMpolicies is only available, if the IBMpolicies support
OID, 1.3.18.0.2.32.18, is present under the ibm-supportedcapabilities of
the root DSE.

c. Within the replicated subtree, in which case the credentials are replicated with the rest of the subtree. Credentials placed in the replicated subtree are created beneath the ibm-replicagroup=default entry for that subtree.

> **Note:** Placing credentials in cn=replication,cn=localhost is considered more secure.

d. Click **Select**.

e. Select the location for the credentials you want to use. Preferably this is cn=replication,cn=localhost.

f. Click **Show credentials**.

g. Expand the list of credentials and select the one you want to use.

h. Click **OK**.

Figure 13-17   Replication schedule and capabilities

9. Specify a replication schedule from the drop-down list or click Add to create one, as shown in Figure 13-17. If you do not specify one it will be default as immediately See Creating replication schedules (page 381).

10. From the list of supplier capabilities, you can deselect any capabilities that you do not want replicated to the consumer.

   – If your network has a mix of servers of different releases, capabilities are available on later releases that are not available on earlier releases. Some

capabilities, like filter ACLs and password policy, make use of operational attributes that are replicated with other changes. In most cases, if these features are used, you want all servers to support them. If all of the servers do not support the capability, you do not want to use it. For example, you would not want different ACLs in effect on each server. However, there might be cases where you might want to use a capability on the servers that support it, and not have changes related to the capability replicated to servers that do not support the capability. In such cases, you can use the capabilities list to mark certain capabilities to not be replicated.

– Click **OK** to create the replica.
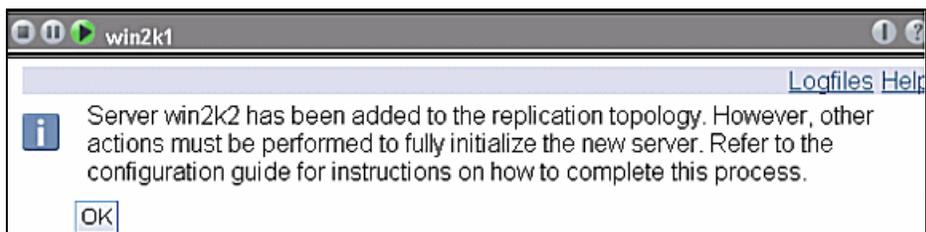
Figure 13-18   Add replica message

11.A message is displayed noting that additional actions must be taken, as shown in Figure 13-18.

– Click **OK**.

– The topology will now look like Figure 13-19.
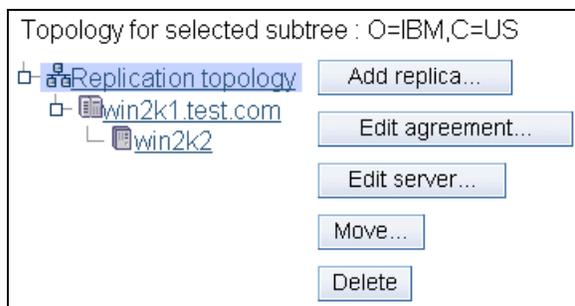
Figure 13-19   Topology after the add

> **Note:** If you are adding more servers as additional replicas or are creating a complex topology, do not proceed with Copying data to the replica (that is, the next step) or Adding the supplier information to the replica until you have finished defining the topology on the master server. If you create the masterfile.ldif after you have completed the topology, it contains the directory entries of the master server and a complete copy of the topology agreements. When you load this file on each of the servers, each server then has the same information.

12. Copy data to the replica.

After creating the replica, you must now export the topology from the master to the replica. This is a manual procedure.

On the master server create an LDIF file for the data. To copy all the data contained on the master server, issue the command:

```
db2ldif -o c:\masterfile.ldif
```

Also you will need to issue the following command to get a copy of the credentials that are in the cn=localhost:

```
db2ldif -o c:\localhost.ldif -s "cn=replication,cn=localhost"
```

The '-s' will allow you to just get the data from a single subtree like `cn=replication,cn=localhost.`

> **Note:** The four operational attributes, createTimestamp, creatorsName, modifiersName, and modifyTimestamp are exported to the LDIF file unless the -j option is specified.

On the machine which you are configuring as the replica:

1. Ensure that the suffixes used by the master are defined in the ibmslapd.conf file.

2. Stop the replica server.

3. Copy both ldif files to the replica and issue the commands:

```
ldif2db -r no -i masterfile.ldif
ldif2db -r no -i localhost.ldif
```

(The '-r no' says not to replicate the data that is loaded.)

The replication agreements, schedules, credentials and entry data are loaded on the replica.

4. Start the server.

Adding supplier information to the replica

You need to change the replica's configuration to identify who is authorized to replicate changes to it, and add a referral to a master.

You can use either of these two options, depending on your situation.

▶ Set the replication bind DN (and password) and a default referral for all subtrees replicated to a server using the 'default credentials and referral'. This might be used when all subtrees are replicated from the same supplier.

▶ Set the replication bind DN and password independently for each replicated subtree by adding supplier information for each subtree. This might be used when each subtree has a different supplier (that is, a different master server for each subtree).

On the machine where you are creating the replica (that is, connect the Web admin tool to the replica server) for normal master and replica replication it is best to use 'default credentials and referral'. We will cover the user of a subtree later in this section.

1. Expand **Replication management** in the navigation area and click **Manage replication properties**, as shown in Figure 13-20.
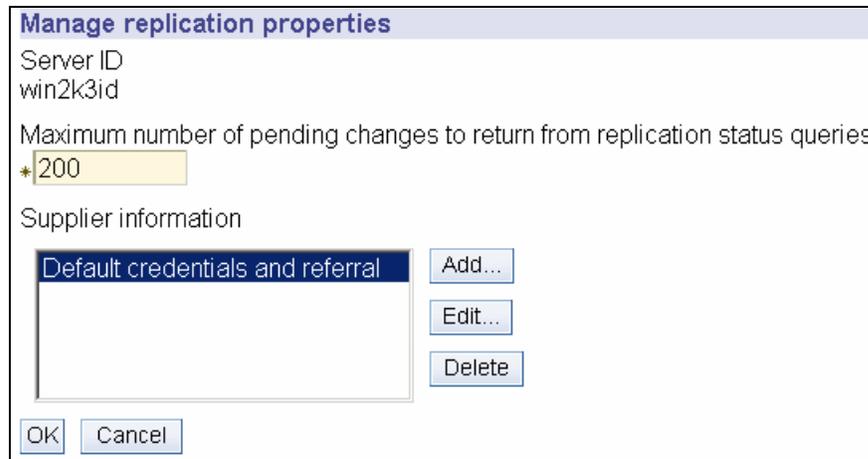
Figure 13-20   Manage replication properties

2. Highlight **default credentials and referral** and click **Edit**, and the window in Figure 13-21 on page 357 is shown.

Figure 13-21 Edit default credentials and referral

3. Enter in the suppliers LDAP URL in this format: ldap://supplier name: port number/.

4. Enter the replication bindDN. In this example, cn=masterbind.

5. Depending on the type of credential, enter and confirm the credential password. (You previously recorded this for future use.)

   – Simple Bind - Specify the DN and password.

   – Kerberos - If the credentials on the supplier do not identify the principal and password, that is, the server's own service principal is to be used, then the bind DN is ibm-kn=ldap/<yourservername@yourrealm>. If the credentials has a principal name such as <myprincipal@myrealm>, use that as the DN. In either case a password in not needed.

   – SSL w/ EXTERNAL bind - Specify the subject DN for the certificate and no password.

6. Click **OK**.

7. You must restart the replica for the changes to take effect.

> **Note:** What this last step really did was to add the following to the
> ibmslapd.conf file:
>
> ```
> dn: cn=Master Server, cn=configuration
> cn: Master Server
> ibm-slapdMasterDN: cn=masterbind
> ibm-slapdMasterPW: >encrypted password<
> ibm-slapdMasterReferral: ldap://win2k1:389/
> objectclass: ibm-slapdConfigEntry
> objectclass: ibm-slapdReplication
> objectclass: top
> ```
>
> This is the reason why you need to re-start the ibmslapd process. This will
> be read in when you bring back up ibmslapd.

8. The replica is in a suspended state by default when you create them and no
   replication is occurring. After you have finished setting up your replication
   topology, you must log onto the Web Admin tool on the Master server and
   click **Manage queues**, the window shown in Figure 13-22 is shown.
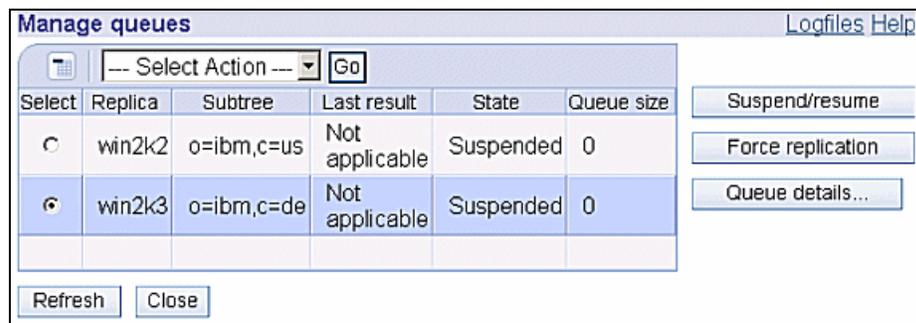
Figure 13-22   Manage queues on the master server

9. Select the replica and click **Suspend/resume** to start replication. It will come
   up in Active state first, as shown in Figure 13-23 on page 359.
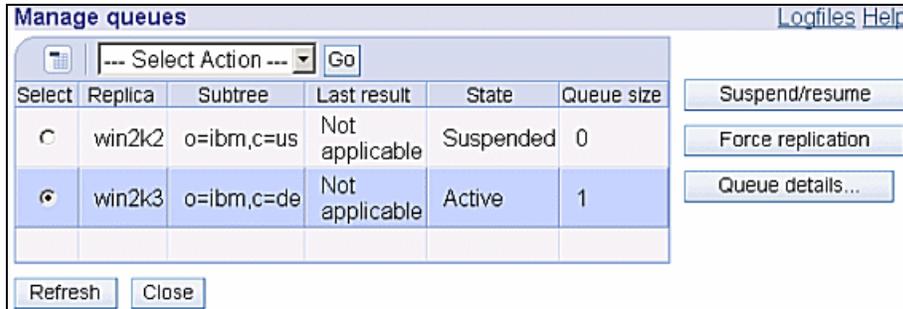
Figure 13-23   Manage queues select replica

10.Click **Queue Details** and the window in Figure 13-24 is shown.
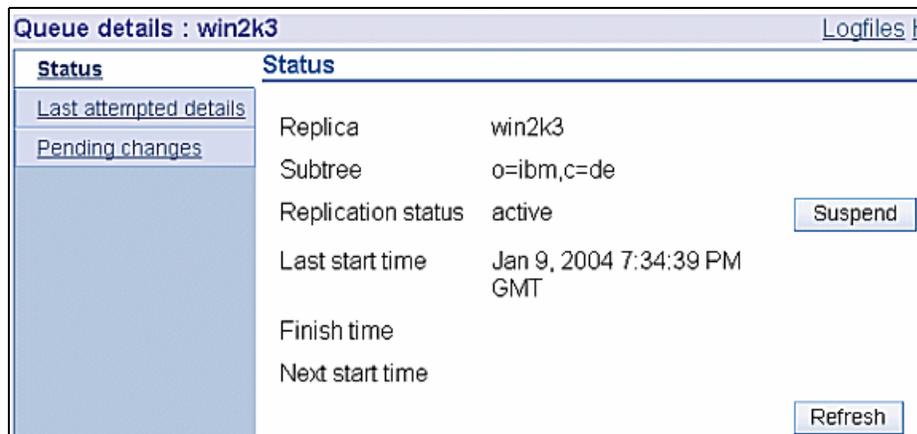
Figure 13-24   Queue details

11.Click **Last attempted details** and then click **Refresh**. This shows `OK`, as shown in Figure 13-25 on page 360

Figure 13-25   Queue status last attempted details

Figure 13-26   Queue details pending changes

12. Pending changes shows the count of '0'. This means that there are no more changes pending, as shown in Figure 13-26.

13. See Manage queues, as shown in Figure 13-27 on page 361, for more detailed information. The replica now receives updates from the master.
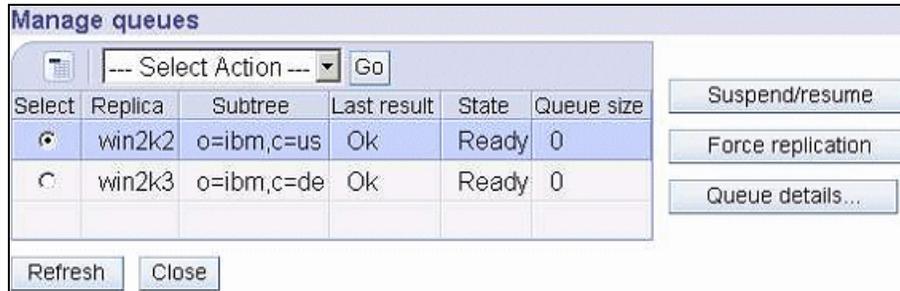
Figure 13-27 Manage queues showing both subtrees replication working

14. Doing the same steps for the other suffix, now both suffixes replication is working with out problems.

13.4.2 Using the command line

> **Note:** As you can tell from using the Web admin tool that it does take time. It is better if you use the Web admin tool for only simple replication scenarios and use command line and LDIF files for any complex replication scenarios.
>
> This scenario assumes that you are creating new replicated subtrees.

Create a LDIF file with the information in Example 13-1 on page 362, named masterreplica.ldif and then load with the following command after you stop the master and replicas:

```
ldif2db -r no -i c:\masterreplica.ldif
```

Load this Into all the servers in the replication before you load customer data. To create one or more replicas for one or more subtrees, you need to create a replica agreement between the master and the replicas. The relationship between the three servers is that the master is the supplier to the two replicas and the replicas are a consumer of the master.

In Example 13-1 on page 362 we used the changed ibm-slapdServerId instead of the ID that would have been generated. The ones we used were win2k1uid, win2k2id, and win2k3id. If you were using the regular ones that were generated on install then you would have to use those in place of the ones we used. This ldif file is built for servers that are ITDS 5.1 and later.

Note: If you are copying a subtree to a IDS 4.1 or earlier server, you must not copy the ibm-replicagroup=default subtree and you must remove the ibm-replicationcontext auxiliary class, because neither of these are supported by the 4.1 schema.

Example 13-1   masterreplica.ldif file

```
###Replication Context - needs to be on all servers in replication
dn: cn=replication,cn=localhost
objectclass=container
objectclass=top
cn=replication

###Replication Credentials - needs to be on all servers in ###replication
agreement
dn: cn=masterbind,cn=replication,cn=localhost
replicaCredentials=secret
description=master bind credential
objectclass=ibm-replicationCredentials
objectclass=ibm-replicationCredentialsSimple
objectclass=top
replicaBindDN=cn=masterbind
cn=masterbind

###New objectclass ibm-replicationContext needs to be attach to each
###subtree / suffix that is replicated and what the replica referral ###URL
will be for that replication
dn: o=ibm,c=us
objectclass: organization
objectclass: top
objectclass: ibm-replicationContext
o: ibm
ibm-replicareferralurl: ldap://win2k1:389

dn: o=ibm,c=de
objectclass: organization
objectclass: top
objectclass: ibm-replicationContext
o: ibm
ibm-replicareferralurl: ldap://win2k1:389

###Replication entry for IBMpolicies
dn: cn=replication,cn=IBMpolicies
objectclass: container
objectclass: top
cn: replication
```

```
###Replica Group for o=ibm,c=us
dn: ibm-replicaGroup=default,o=ibm,c=us
ibm-replicaGroup: default
objectclass: ibm-replicaGroup
objectclass: top

###Replica SubEntry for o=ibm,c=us
dn: cn=win2k1.test.com,ibm-replicaGroup=default,o=ibm,c=us
objectclass: ibm-replicaSubentry
objectclass: top
ibm-replicaServerId: win2k1uid
ibm-replicationServerIsMaster: TRUE
cn: win2k1.test.com

###Replica Group for o=ibm,c=de
dn: ibm-replicaGroup=default,o=ibm,c=de
ibm-replicaGroup: default
objectclass: ibm-replicaGroup
objectclass: top

###Replica SubEntry for o=ibm,c=de
dn: cn=win2k1.test.com,ibm-replicaGroup=default,o=ibm,c=de
objectclass: ibm-replicaSubentry
objectclass: top
ibm-replicaServerId: win2k1uid
ibm-replicationServerIsMaster: TRUE
cn: win2k1.test.com

###Replication Agreement to Replica Sever for o=ibm,c=us
dn: cn=win2k2,cn=win2k1.test.com,ibm-replicaGroup=default,O=IBM,C=US
ibm-replicaConsumerId: win2k2id
ibm-replicationOnHold: TRUE
ibm-replicaCredentialsDN: cn=masterbind,cn=replication,cn=localhost
ibm-replicaURL: ldap://win2k2:389
description: Win 2k 2 server
objectclass: ibm-replicationAgreement
objectclass: top
cn: win2k2

###Replication Agreement to Replica Sever for o=ibm,c=de
dn: cn=win2k3,cn=win2k1.test.com,ibm-replicaGroup=default,O=IBM,C=DE
ibm-replicaConsumerId: win2k3id
ibm-replicationOnHold: TRUE
ibm-replicaCredentialsDN: cn=masterbind,cn=replication,cn=localhost
ibm-replicaURL: ldap://win2k3:389
description: win 2k 3 replica
objectclass: ibm-replicationAgreement
objectclass: top
```

```
cn: win2k3
```

Add the following to the ibmslapd.conf files or the replicas:

```
dn: cn=Master Server, cn=configuration
cn: Master Server
ibm-slapdMasterDN: cn=masterbind
ibm-slapdMasterPW: secret
ibm-slapdMasterReferral: ldap://win2k1:389/
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdReplication
objectclass: top
```

When you have done these steps you can start all the servers and test out to see if replication is working. The best way to do this is to bring up the Web Administration tool and check manage queues, as shown in Figure 13-27 on page 361.

13.4.3 Promoting a replica to peer/master

The below scenario describes the steps required to promote a replica to master so that it becomes a peer to its former master. In order to configure a server as a peer to a given server, it has to be added as a replica to the master and then promoted to a peer as described below. We will take what we did with the scenario we just finish working with and now make them peer to peer for each suffix.

1. Connect the Web Administration Tool to the master and click **Replication Management**.

2. Select the appropriate subtree from the right hand panel and click **Show topology**. The replication topology for the given subtree is displayed in Figure 13-28 on page 365.
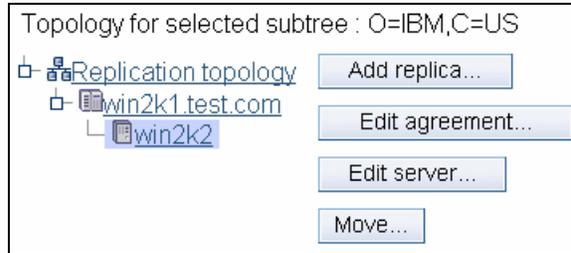
Figure 13-28   Show topology

> **Note:** The replica that you want to promote to a peer should not have other replicas configured below it.

3. Select the appropriate replica you want to promote to a peer from the replication topology and click **Move**.
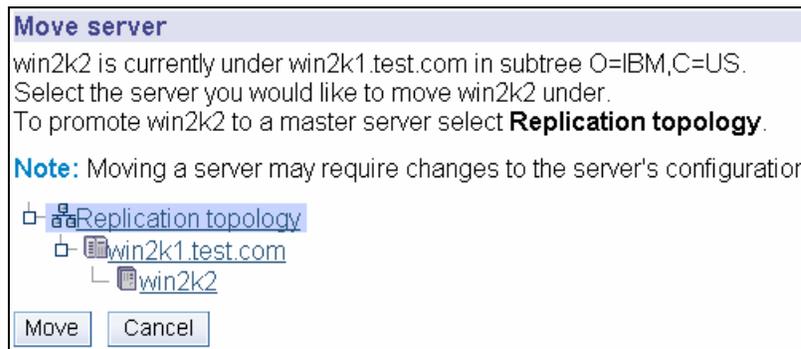
Figure 13-29   Move server

4. On the screen that appears (Figure 13-29), Replication topology is highlined by default. Take the default and click **Move**.
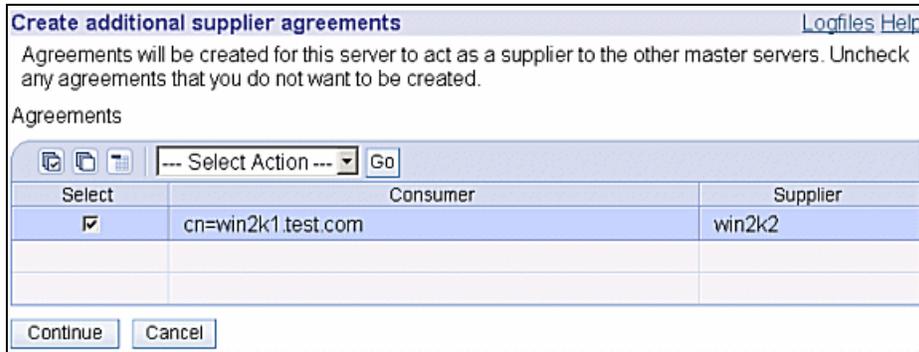
Figure 13-30   Additional supplier agreements

5. The next screen (Figure 13-30) asks for agreements to be created from the newly promoted peer to the existing masters and replicas in the topology. It will default with a checkmark on the one that you will make peer to peer. Click **Continue**.
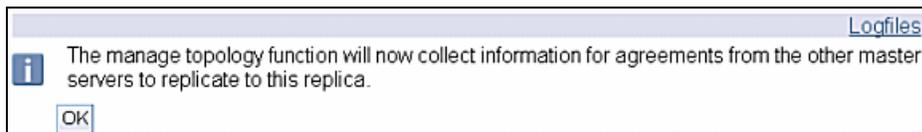
Figure 13-31   Move message

6. This screen will come up to inform you what is going to happen (Figure 13-31). Click **OK**.

Figure 13-32   Select credential

7. You will need to click **Add Credentials** with the radio button on `o=ibm,c=us`, as shown in Figure 13-32.

Figure 13-33   Authentication method

8. Fill in Credential name for example, `cn=peeribmus` and keep it Simple Bind, as shown in Figure 13-33.

9. Click **Next**.

Figure 13-34   Simple bind

10. Fill in the bind DN, that is, cn=peeribmus and the Bind password, that is, secret. Enter the same password to confirm. You can put in a description if you want, as shown in Figure 13-34.

11. Click **Finish**.

Figure 13-35   Select credential

12. Click the down arrow and pick the credential you just made (**peeribmus**), as shown in Figure 13-35.

13. Click **OK**.
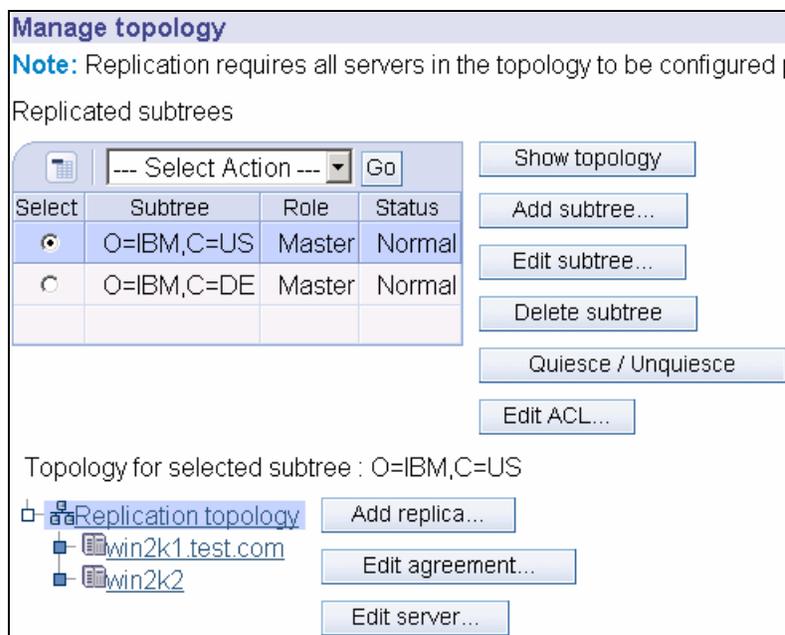
Figure 13-36   Manage topology

14. The screen in Figure 13-36, shows that the replica has been promoted to a peer.

15. The next step is to build the supplier (win2k2) information on the new replica (win2k1) Click **Add** and pick **o=ibm,c=us**. Then click **OK**, as shown in Figure 13-37 on page 370.

.

Figure 13-37   Manage replication properties on master server

Figure 13-38   Supplier credentials

16. Enter the bind DN that you created on the supplier, that is, cn=peeribmus and enter bind password twice, that is, secret, as shown in Figure 13-38.

17. Click **OK**.

18. Restart the replica to have it take affect.

19. Log on to the Web Admin Tool for Win2k2 the new supplier and click **Manage queues**, as shown in Figure 13-39 on page 371.
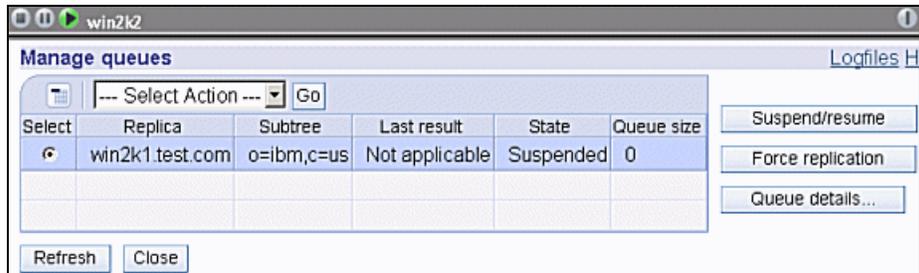
Figure 13-39 Manage queues for Win2k2 supplier

20. Click **Suspend/resume** to start up the replication. Buy default it will come up in Suspended state, as shown in Figure 13-40.
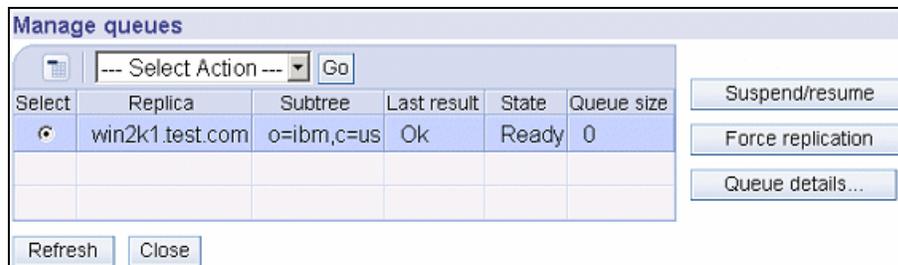
Figure 13-40 Manage queues

21. Do the same steps for the other suffix on Win2k3 for o=ibm,c=de, as shown in Figure 13-41.

Figure 13-41 Topology for o=ibm,c=de

13.4.4 Command line for a complex replication

For any Complex Replication agreements it is best that you do it by command line and load individual LDIF loads. This way you can lay everything out first. It will take a very long time and more work to do it with the Web Admin Tool.

For this scenario we will be using only one bind credential that will be under cn=localhost. There are two subtree replications. There is one master server that has both subtree fully loaded and one of the two subtrees has two more peer masters and the other subtree has two more peer masters for a total of five servers.

Example 13-2   LDIF file for complex replication setup

```
dn: cn=replication,cn=localhost
objectclass: container
objectclass: top
cn: replication

###Replication Group
dn: ibm-replicaGroup=default,ou=people,o=ibm,c=us
objectclass: top
objectclass: ibm-replicaGroup
ibm-replicaGroup: default

###Bind Credentials/method to Peer server - replication agreement
```

```
###points to this.
dn: cn=ReplicationCreds,cn=replication,cn=localhost
objectclass: ibm-replicationCredentialsSimple
cn: ReplicationCreds
replicaBindDN: cn=master
replicaCredentials: master
description: Bindmethod of master to Peer1

###Master SubEntry for ou=people,o=ibm,c,us
dn: cn=win2k1.test.com,ibm-replicaGroup=default,ou=people,o=ibm,c=us
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: win2k1
ibm-replicationServerIsMaster: true
cn: masterpeer
description: masterpeer server

### Peer2 Subentry
dn: cn=win2k2.test.com,ibm-replicaGroup=default,ou=people,o=ibm,c=us
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: win2k2
ibm-replicationServerIsMaster: true
cn: peer2
description: peer2 server

### Peer3 SubEntry
dn: cn=win2k3.test.com,ibm-replicaGroup=default,ou=people,o=ibm,c=us
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: win2k3
ibm-replicationServerIsMaster: true
cn: peer3
description: peer3 server

###Masterpeer to peer2 agreement
dn:
cn=peer2,cn=win2k1.test.com,ibm-replicaGroup=default,ou=people,o=ibm,c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: peer2
ibm-replicaConsumerId: win2k2
ibm-replicaUrl: ldap://win2k2:389
ibm-replicaCredentialsDN: cn=ReplicationCreds,cn=replication,cn=localhost
description: Masterpeer to peer2 server

###Masterpeer to peer3 agreement
dn:
cn=peer3,cn=win2k1.test.com,ibm-replicaGroup=default,ou=people,o=ibm,c=us
```

```
objectclass: top
objectclass: ibm-replicationAgreement
cn: peer3
ibm-replicaConsumerId: win2k3
ibm-replicaUrl: ldap://win2k3:389
ibm-replicaCredentialsDN: cn=ReplicationCreds,cn=replication,cn=localhost
description: Masterpeer to peer3 server

###peer2 to Master agreement
dn:
cn=masterpeer,cn=win2k2test.com,ibm-replicaGroup=default,ou=people,o=ibm,c=
us
objectclass: top
objectclass: ibm-replicationAgreement
cn: Masterpeer
ibm-replicaConsumerId: win2k1
ibm-replicaUrl: ldap://win2k1:389
ibm-replicaCredentialsDN: cn=ReplicationCreds,cn=replication,cn=localhost
description: Peer 2 to Masterpeer server

###peer2 to Peer3 agreement
dn:
cn=peer3,cn=win2k2test.com,ibm-replicaGroup=default,ou=people,o=ibm,c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: peer3
ibm-replicaConsumerId: win2k3
ibm-replicaUrl: ldap://win2k3:389
ibm-replicaCredentialsDN: cn=ReplicationCreds,cn=replication,cn=localhost
description: Peer 2 to Peer 3 server

###peer3 to Master agreement
dn:
cn=masterpeer,cn=win2k3.test.com,ibm-replicaGroup=default,ou=people,o=ibm,c
=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: Masterpeer
ibm-replicaConsumerId: win2k1
ibm-replicaUrl: ldap://win2k1:389
ibm-replicaCredentialsDN: cn=ReplicationCreds,cn=replication,cn=localhost
description: Peer 3 to Masterpeer server

###peer3 to Peer2 agreement
dn:
cn=peer2,cn=win2k3.test.com,ibm-replicaGroup=default,ou=people,o=ibm,c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: peer2
```

```
ibm-replicaConsumerId: win2k2
ibm-replicaUrl: ldap://win2k2:389
ibm-replicaCredentialsDN: cn=ReplicationCreds,cn=replication,cn=localhost
description: Peer 3 to Peer 2 server


###Replication Group
dn: ibm-replicaGroup=default,ou=app,o=ibm,c=us
objectclass: top
objectclass: ibm-replicaGroup
ibm-replicaGroup: default


###Master SubEntry ou=app,o=ibm,c=us
dn: ibm-replicaServerId=win2k1,ibm-replicaGroup=default,ou=app,o=ibm,c=us
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: win2k1
ibm-replicationServerIsMaster: true
cn: master
description: master server


###Peer 4 SubEntry
dn: ibm-replicaServerId=win2k4,ibm-replicaGroup=default,ou=app,o=ibm,c=us
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: win2k4
ibm-replicationServerIsMaster: true
cn: peer4
description: Peer 4 server


### Peer5 Subentry
dn: cn=win2k5.test.com,ibm-replicaGroup=default,ou=app,o=ibm,c=us
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: win2k5
ibm-replicationServerIsMaster: true
cn: peer5
description: peer1 server


###Master to peer5 agreement
dn: cn=win2k5,cn=win2k1.test.com,ibm-replicaGroup=default,ou=app,o=ibm,c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: peer5
ibm-replicaConsumerId: win2k5
ibm-replicaUrl: ldap://win2k5:389
ibm-replicaCredentialsDN: cn=ReplicationCreds,cn=replication,cn=localhost
description: master to peer5 server


###peer5 to master agreement
```

```
dn: cn=win2k1,cn=win2k5.test.com,ibm-replicaGroup=default,ou=app,o=ibm,c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: master
ibm-replicaConsumerId: win2k1
ibm-replicaUrl: ldap://win2k1:389
ibm-replicaCredentialsDN: cn=ReplicationCreds,cn=replication,cn=localhost
description: peer5 to master server

###master to peer4 agreement
dn: cn=win2k4,cn=win2k1.test.com,ibm-replicaGroup=default,ou=app,o=ibm,c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: peer4
ibm-replicaConsumerId: win2k4
ibm-replicaUrl: ldap://win2k4:389
ibm-replicaCredentialsDN: cn=ReplicationCreds,cn=replication,cn=localhost
description: master to peer4 server

###peer4 to master agreement
dn: cn=win2k1,cn=win2k4.test.com,ibm-replicaGroup=default,ou=app,o=ibm,c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: master
ibm-replicaConsumerId: win2k4
ibm-replicaUrl: ldap://win2k4:389
ibm-replicaCredentialsDN: cn=ReplicationCreds,cn=replication,cn=localhost
description: peer4 to master server

###Peer4 to peer5 agreement
dn: cn=win2k5,cn=win2k4.test.com,ibm-replicaGroup=default,ou=app,o=ibm,c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: peer4
ibm-replicaConsumerId: win2k5
ibm-replicaUrl: ldap://win2k5:389
ibm-replicaCredentialsDN: cn=ReplicationCreds,cn=replication,cn=localhost
description: peer4 to peer5 server

###peer5 to peer4 agreement
dn: cn=win2k4,cn=win2k5.test.com,ibm-replicaGroup=default,ou=app,o=ibm,c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: peer5
ibm-replicaConsumerId: win2k4
ibm-replicaUrl: ldap://win2k4:389
ibm-replicaCredentialsDN: cn=ReplicationCreds,cn=replication,cn=localhost
description: peer5 to peer4 server one
```

Add the following to the ibmslapd.conf files for all five servers:

```
dn: cn=Master Server, cn=configuration
cn: Master Server
ibm-slapdMasterDN: cn=masterbind
ibm-slapdMasterPW: secret
ibm-slapdMasterReferral: ldap://win2k1:389/
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdReplication
objectclass: top
```

13.5 Web administration tasks for managing replication

This part of the section covers tasks that have not be covers elsewhere in this section on Replication that can done using the Web Administration GUI.

13.5.1 Managing topology

In this section we discuss managing topology.

Editing an agreement

From the Replication topology displayed, select a replication agreement by clicking it. Then click **Edit agreement**. You can change the following information for the replica:

► On the Server tab you can only change:

  – Hostname
  – Port
  – Enable SSL
  – Description

► On the Additional tab you can change:

  – Credentials.

  – Replication schedules.

  – Change the capabilities replicated to the consumer replica. From the list of supplier capabilities, you can deselect any capabilities that you do not want replicated to the consumer.

Editing a server

Note: A gateway server must be an IBM Tivoli Directory Server Version 5.2 server or an IBM Directory Server Version 5.1 server with a fix pack (FP1 or later) that supports gateway replication.

You can designate whether a master server is to have the role of a gateway server in the replication site.

To designate a master as a gateway server, select the **Server is a gateway** check box.

To remove the role of a gateway server from a master server, deselect the Server is a gateway check box.

Demoting a master server

To change the role of a server from a master to a replica do the following:

1. Connect the Web Administration Tool to the server that you want to demote.

2. Click **Manage topology**.

3. Select the subtree and click **Show topology**.

4. Delete all the agreements for the server you want to demote.

5. Select the server you are demoting and click **Move**.

6. Select the server under which you are going to place the demoted server and click **Move**.

7. Just as you would for a new replica, create new supplier agreements between the demoted server and its supplier.

Replicating a subtree

To replicate a subtree, expand the Replication management category in the navigation area and click **Manage topology**, and perform the following steps:

1. Click **Add subtree**.

2. Enter the DN of the subtree that you want to replicate or click **Browse** to expand the entries to select the entry that is to be the root of the subtree.

3. Enter the master server referral URL. This must be in the form of an LDAP URL, for example ldap://<myservername>.<mylocation>.<mycompany>.com.

4. Click **OK**.

5. The new subtree is displayed on the Manage topology panel under the heading Replicated subtrees.

> **Note:** On the Linux, Solaris, and HP-UX platforms, if a referral fails, ensure that the environment variable LDAP_LOCK_REC has been set in your system environment. No specific value is required.
>
> ```
> set LDAP_LOCK_REC=anyvalue
> ```

Editing a subtree

Use this option to change the URL of the master server that this subtree and its replicas send updates to. You need do this if you change the port number or host name of the master server, change the master to a different server. To edit a subtree perform the following steps:

1. Select the subtree you want to edit.

2. Click **Edit** subtree.

3. Enter the master server referral URL. This must be in the form of an LDAP URL, for example
   ldap://<mynewservername>.<mylocation>.<mycompany>.com.

Depending on the role being played by the server on this subtree (whether it is a master, replica or forwarder), different labels and buttons appear on the panel.

▶ When the subtree's role is replica, a label indicating that the server acts as a replica or forwarder is displayed along with the button Make server a master. If this button is clicked then the server which the Web Administration Tool is connected to becomes a master.

▶ When the subtree is configured for replication only by adding the auxiliary class (no default group and subentry present), then the label `This subtree is not replicated` is displayed along with the button `Replicate subtree`. If this button is clicked, the default group and the subentry is added so that the server with which the Web Administration Tool is connected to becomes a master.

▶ If no subentries for the master servers are found, the label `No master server is defined for this subtree` is displayed along with the button titled `Make server a master`. If this button is clicked, the missing subentry is added so that the server with which the Web Administration Tool is connected to becomes a master.

Removing a subtree

To remove a subtree:

1. Select the subtree you want to remove

2. Click **Delete subtree**.

3. When asked to confirm the deletion, click **OK**.

4. The subtree is removed from the **Replicated subtree** list.

> **Note:** This operation succeeds only if the ibm-replicaGroup=default is entry is empty.

Quiescing the subtree

This function is useful when you want to perform maintenance on or make changes to the topology. It minimizes the number of updates that can be made to the server. A quiesced server does not accept client requests. It accepts requests only from an administrator using the Server Administration control. This function is Boolean. To quiesce or unquiesce the subtree perform the following steps:

1. Click **Quiesce/Unquiesce** to quiesce the subtree.

2. When asked to confirm the action, click **OK**.

3. Click **Quiesce/Unquiesce** to unquiesce the subtree.

4. When asked to confirm the action, click **OK**.

Editing access control lists

Replication information (replica subentries, replication agreements, schedules, possibly credentials) are stored under a special object, ibm-replicagroup=default. The ibm-replicagroup object is located immediately beneath the root entry of the replicated subtree. By default, this subtree inherits ACL from the root entry of the replicated subtree. This ACL might not be appropriate for controlling access to replication information.

Required authorities:

► Control replication - You must have write access to the ibm-replicagroup=default object (or be the owner/administrator).

► Cascading control replication - You must have write access to the ibm-replicagroup=default object (or be the owner/administrator).

► Control queue - You must have write access to the replication agreement.

13.5.2 Modifying replication properties

Expand the **Replication management** category in the navigation area and click **Manage replication properties**. On this panel you can:

► Change the maximum number of pending changes to return from replication status queries. The default is 200.

► Add, edit, or delete supplier information. Click **OK**. The subtree of the supplier is added to the Supplier information list.

Editing supplier information

To edit the supplier information, perform the following:

1. Select the supplier subtree that you want to edit.

2. Click **Edit**.

3. If you are editing Default credentials and referral, which is used to create the cn=Master Server entry under cn=configuration, enter the URL of the server from which the client wants to receive replica updates in the Default supplier's LDAP URL field. This needs to be a valid LDAP URL (ldap://). Otherwise, skip to step 4.

4. Enter the replication bind DN for the new credentials you want to use.

5. Enter and confirm the credential password.

6. Click **OK**.

Removing supplier information

To remove the supplier information, perform the following steps:

1. Select the supplier subtree that you want to remove.

2. Click **Delete**.

3. When asked to confirm the deletion, click **OK**.

4. The subtree is removed from the Supplier information list.

13.5.3 Creating replication schedules

By default, the changes made on the master/peer server are replicated immediately. But, we can define an optional replication schedule object and attach it to a replication agreement in order to allow replication of changes at given times in a day and given day in a week. Schedules are defined per replication agreement.

Expand the **Replication management** category in the navigation area and click **Manage schedules**.

Creating a daily schedule

In this section we discuss creating a daily schedule.

Figure 13-42   Add daily schedule

On the Daily schedule tab, select the subtree for which you want to create the schedule and click **Show schedules**. If any schedules exist, they are displayed in the Daily schedules box. To create or add a new schedule, refer to Figure 13-42, and perform the following steps:

1. Click **Add**. Enter a name for the schedule. For example monday1.

2. Select the time zone setting, either UTC or local.

3. Select a replication type from the drop-down menu:

   – Immediate

   Performs any pending entry updates since the last replication event and then updates entries continuously until the next scheduled update event is reached.

   – Once

   Performs all pending updates prior to the starting time. Any updates made after the start time wait until the next scheduled replication event.

   Select a start time for the replication event.

4. Click **Add**. The replication event type and time are displayed.

   a. Add or remove events to complete your schedule. The list of events is refreshed in chronological order.

b. When you are finished, click **OK**.

> **Note:** If replication events are scheduled too closely together, a replication event might be missed if the updates from the previous event are still in progress when the next event is scheduled

c. You can select a day and click **Add a daily schedule** to create a daily replication schedule for it. If you create a daily schedule it becomes the default schedule for each day of the week. You can:

   •   Keep the daily schedule as the default for each day or select a specific day and change the schedule back to none. Remember that the last replication event that occurred is still in effect for a day that has no replication events scheduled.

   •   Modify the daily schedule by selecting a day and clicking **Edit a daily schedule**. Remember changes to a daily schedule affect all days using that schedule, not just the day you selected.

   •   Create a different daily schedule by selecting a day and clicking **Add a daily schedule**. After you have created this schedule it is added to the Daily schedule drop-down menu. You must select this schedule for each day that you want the schedule to be used.

d. When you are finished, click **OK**.

Creating a weekly schedule

In this section we discuss how to create a weekly schedule.

Figure 13-43   Add weekly schedule

On the Weekly schedule tab, select the subtree for which you want to create the schedule and click **Show schedules**. If any schedules exist, they are displayed in the Weekly schedules box. To create or add a new schedule, refer to Figure 13-43, and perform the following steps:

1. Click **Add**.

2. Enter a name for the schedule. For example schedule1.

3. For each day, Sunday through Saturday, the daily schedule is specified as None. This means that no replication update events are scheduled. The last replication event, if any, is still in effect. Because this is a new replica, there are no prior replication events, therefore, the schedule defaults to immediate replication.

13.5.4 Managing queues

This task allows you to monitor status of replication for each replication agreement (queue) used by this server.

Expand the **Replication management** category in the navigation area and click **Manage queues**.

1. Select the replica for which you want to manage the queue.

2. Depending on the status of the replica, you can click **Suspend/resume** to stop or start replication.

3. Click **Force replication** to replicate all the pending changes regardless of when the next replication is scheduled.

4. Click **Queue details**, for more complete information about the replica's queue. You can also manage the queue from this selection.

5. Click **Refresh** to update the queues and clear server messages.

Queue details

If you clicked **Queue details**, three tabs are displayed:

► Status
► Last attempted details
► Pending changes

The Status tab displays the replica name, its subtree, its status, and a record of replication times. From this panel you can suspend or resume replication by clicking **Resume**. Click **Refresh** to update the queue information.

The Last attempted details tab gives information about the last update attempt. If an entry is not able to be loaded click **Skip blocking entry** to continue replication with the next pending entry. Click **Refresh** to update the queue information.

The Pending changes tab shows all the pending changes to the replica. If replication is blocked you can delete all the pending changes by clicking **Skip all**. Click **Refresh** to update the list of pending changes to reflect any new update or updates that have been processed.

13.6 Repairing replication differences between replicas

This section discusses ways to repair replication differences between replicas, and provides two examples, Example 13-3 on page 392 and Example 13-4 on page 393.

13.6.1 The ldapdiff command tool

LDAPDIFF is the LDAP replica synchronization tool. If you find that you have a replica that might be out of sync with the master you can use this process to sync up the replica to match the master for both data and schema.

Synopsis

This shows how to use the ldapdiff to test to see if there are differences between master and replicas data:

```
ldapdiff -b baseDN -sh host -ch host [-a] [-C countnumber]
[-cD dn] [-cK keyStore] [-cw password] -[cN keyStoreType]
[-cp port] [-cP keyStorePwd] [-ct trustStoreType] [-cT trustStore]
[-cY trustStorePwd] [-cZ] [-F] [-j] [-L filename] [-sD dn]
[-sK keyStore] [-sw password] -[sN keyStoreType] [-sp port]
[-sP keyStorePwd] [-st trustStoreType] [-sT trustStore]
[-sY trustStorePwd] [-sZ] [-v]
```

This shows how to use the ldapdiff to test to see if there are differences between master and replicas Schema:

```
ldapdiff -S -sh host -ch host [-a] [-C countnumber][-cD dn]
[-cK keyStore] [-cw password] -[cN keyStoreType] [-cp port]
[-cP keyStorePwd] [-ct trustStoreType] [-cT trustStore]
[-cY trustStorePwd] [-cZ] [-j][-L filename] [-sD dn]
[-sK keyStore] [-sw password] [-sN keyStoreType] [-sp port]
[-sP keyStorePwd] [-st trustStoreType] [-sT trustStore]
[-sY trustStorePwd] [-sZ] [-v]
```

Description

This tool synchronizes a replica server with its master. To display syntax help for ldapdiff, type:

```
ldapdiff -?
```

Options

The following options apply to the ldapdiff command. There are two subgroupings that apply specifically to either the supplier server or the consumer server.

► -a - Specifies to use server administration control for writes to a read-only replica.

► -b baseDN - Use searchbase as the starting point for the search instead of the default. If -b is not specified, this utility examines the LDAP_BASEDN environment variable for a searchbase definition.

► -C countnumber - Counts the number of entries to fix. If more than the specified number of mismatches are found, the tool exits.

► -F - This is the fix option. If specified, content on the consumer replica is modified to match the content of the supplier server. This cannot be used if the -S is also specified.

► -j - Indicates to ignore the operational attributes in the LDIF file.

- ► -L - If the -F option is not specified, use this option to generate an LDIF file for output. The LDIF file can be used to update the consumer to eliminate the differences.

- ► -S - Specifies to compare the schema on both of the servers.

- ► -v - Use verbose mode, with many diagnostics written to standard output.

Options for a replication supplier: The following options apply to the consumer server and are denoted by an initial fsf in the option name.

- ► -sD - dn Use dn to bind to the LDAP directory. dn is a string-represented DN.

- ► -sh - host Specifies the host name.

- ► -sK keyStore - Specify the name of the SSL key database file with default extension of kdb. If the key database file is not in the current directory, specify the fully-qualified key database filename. If a key database filename is not specified, this utility will first look for the presence of the SSL_KEYRING environment variable with an associated filename. If the SSL_KEYRING environment variable is not defined, the default keyring file will be used, if present.

A default keyring file that is, ldapkey.kdb, and the associated password stash file that is, ldapkey.sth, are installed in the /lib directory under LDAPHOME, where LDAPHOME is the path to the installed LDAP support. LDAPHOME varies by operating system platform:

- ► AIX operating systems - /usr/ldap
- ► HP-UX operating systems - /usr/IBMldap
- ► Linux operating systems - /usr/ldap
- ► Solaris operating systems - /opt/IBMldaps
- ► Windows operating systems - c:\Program Files\IBM\LDAP

> **Note:** This is the default install location. The actual LDAPHOME is determined during installation.

See *IBM Directory C-Client SDK Programming Reference* (available at: http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html) for more information about default key database files, and default Certificate Authorities.

If a keyring database file cannot be located, a hard-coded set of default trusted certificate authority roots is used. The key database file typically contains one or more certificates of certificate authorities (CAs) that are trusted by the client. These types of X.509 certificates are also known as trusted roots. For more

information on managing an SSL key database, This parameter effectively enables the -sZ switch.

► -sN keyStoreType - Specify the label associated with the client certificate in the key database file. If the LDAP server is configured to perform server authentication only, a client certificate is not required. If the LDAP server is configured to perform client and server Authentication, a client certificate might be required. keyStoreType is not required if a default certificate/private key pair has been designated as the default. Similarly, keyStoreType is not required if there is a single certificate/private key pair in the designated key database file. This parameter is ignored if neither -sZ nor -sK is specified.

► -sp ldapport - Specify an alternate TCP port where the ldap server is listening. The default LDAP port is 389. If -sp is not specified and -sZ is specified, the default LDAP SSL port 636 is used.

► -sP keyStorePwd - Specify the key database password. This password is required to access the encrypted information in the key database file, which may include one or more private keys. If a password stash file is associated with the key database file, the password is obtained from the password stash file, and the -sP parameter is not required. This parameter is ignored if neither -sZ nor -sK is specified.

► -st trustStoreType - Specify the label associated with the client certificate in the trust database file. If the LDAP server is configured to perform server authentication only, a client certificate is not required. If the LDAP server is configured to perform client and server Authentication, a client certificate might be required. trustStoreType is not required if a default certificate/private key pair has been designated as the default. Similarly, trustStoreType is not required if there is a single certificate/private key pair in the designated key database file. This parameter is ignored if neither -sZ nor -sT is specified.

► -sT trustStore - Specify the name of the SSL trust database file with default extension of tdb. If the trust database file is not in the current directory, specify the fully-qualified trust database filename. If a trust database filename is not specified, this utility will first look for the presence of the SSL_KEYRING environment variable with an associated filename. If the SSL_KEYRING environment variable is not defined, the default keyring file will be used, if present.

A default keyring file that is, ldapkey.tdb, and the associated password stash file that is, ldapkey.sth, are installed in the /lib directory under LDAPHOME, where LDAPHOME is the path to the installed LDAP support. LDAPHOME varies by operating system platform:

► AIX operating systems - /usr/ldap
► HP-UX operating systems - /usr/IBMldap
► Linux operating systems - /usr/ldap

- Solaris operating systems - /opt/IBMldaps
- Windows operating systems - c:\Program Files\IBM\LDAP

> **Note:** This is the default install location. The actual LDAPHOME is determined during installation.

See *IBM Directory C-Client SDK Programming Reference* (available at: http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html) for more information about default key database files, and default Certificate Authorities.

If a keyring database file cannot be located, a hard-coded set of default trusted certificate authority roots is used. The key database file typically contains one or more certificates of certificate authorities (CAs) that are trusted by the client. These types of X.509 certificates are also known as trusted roots. For more information on managing an SSL key database,

This parameter effectively enables the -sZ switch.

- -sw password | ? - Use password as the password for authentication. Use the ? to generate a password prompt. Using this prompt prevents your password from being visible through the ps command.
- -sY - The password for the trusted database.
- -sZ - Use a secure SSL connection to communicate with the LDAP server. The -Z option is only supported when the SSL component entry, as provided by IBM's GSKit, is installed.

Options for a replication consumer

The following options apply to the consumer server and are denoted by an initial fcf in the option name.

- -cD dn - Use dn to bind to the LDAP directory. dn is a string-represented DN.
- -ch host - Specifies the host name.
- -cK keyStore - Specify the name of the SSL key database file with default extension of kdb. If the key database file is not in the current directory, specify the fully-qualified key database filename. If a key database filename is not specified, this utility will first look for the presence of the SSL_KEYRING environment variable with an associated filename. If the SSL_KEYRING environment variable is not defined, the default keyring file will be used, if present.

A default keyring file that is, ldapkey.kdb, and the associated password stash file that is, ldapkey.sth, are installed in the /lib directory under LDAPHOME, where

LDAPHOME is the path to the installed LDAP support. LDAPHOME varies by operating system platform:

► AIX operating systems - /usr/ldap
► HP-UX operating systems - /usr/IBMldap
► Linux operating systems - /usr/ldap
► Solaris operating systems - /opt/IBMldaps
► Windows operating systems - c:\Program Files\IBM\LDAP

> **Note:** This is the default install location. The actual LDAPHOME is determined during installation.

See *IBM Directory C-Client SDK Programming Reference* (available at: http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html) for more information about default key database files, and default Certificate Authorities.

If a keyring database file cannot be located, a hard-coded set of default trusted certificate authority roots is used. The key database file typically contains one or more certificates of certificate authorities (CAs) that are trusted by the client. These types of X.509 certificates are also known as trusted roots. For more information on managing an SSL key database,

This parameter effectively enables the -cZ switch.

► -cN keyStoreType - Specify the label associated with the client certificate in the key database file. If the LDAP server is configured to perform server authentication only, a client certificate is not required. If the LDAP server is configured to perform client and server Authentication, a client certificate might be required. keyStoreType is not required if a default certificate/private key pair has been designated as the default. Similarly, keyStoreType is not required if there is a single certificate/private key pair in the designated key database file. This parameter is ignored if neither -cZ nor -cK is specified.

► -cp ldapport - Specify an alternate TCP port where the ldap server is listening. The default LDAP port is 389. If -cp is not specified and -cZ is specified, the default LDAP SSL port 636 is used.

► -cP keyStorePwd - Specify the key database password. This password is required to access the encrypted information in the key database file, which may include one or more private keys. If a password stash file is associated with the key database file, the password is obtained from the password stash file, and the -cP parameter is not required. This parameter is ignored if neither -cZ nor -cK is specified.

► -ct trustStoreType - Specify the label associated with the client certificate in the trust database file. If the LDAP server is configured to perform server authentication only, a client certificate is not required. If the LDAP server is

configured to perform client and server Authentication, a client certificate might be required. trustStoreType is not required if a default certificate/private key pair has been designated as the default. Similarly, trustStoreType is not required if there is a single certificate/private key pair in the designated key database file. This parameter is ignored if neither -cZ nor -cT is specified.

► -cT trustStore - Specify the name of the SSL trust database file with default extension of tdb. If the trust database file is not in the current directory, specify the fully-qualified trust database filename. If a trust database filename is not specified, this utility will first look for the presence of the SSL_KEYRING environment variable with an associated filename. If the SSL_KEYRING environment variable is not defined, the default keyring file will be used, if present.

A default keyring file that is, ldapkey.tdb, and the associated password stash file that is, ldapkey.sth, are installed in the /lib directory under LDAPHOME, where LDAPHOME is the path to the installed LDAP support. LDAPHOME varies by operating system platform:

► AIX operating systems - /usr/ldap
► HP-UX operating systems - /usr/IBMldap
► Linux operating systems - /usr/ldap
► Solaris operating systems - /opt/IBMldaps
► Windows operating systems - c:\Program Files\IBM\LDAP

> **Note:** This is the default install location. The actual LDAPHOME is determined during installation.

See *IBM Directory C-Client SDK Programming Reference* (available at: http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html) for more information about default key database files, and default Certificate Authorities.

If a keyring database file cannot be located, a hard-coded set of default trusted certificate authority roots is used. The key database file typically contains one or more certificates of certificate authorities (CAs) that are trusted by the client. These types of X.509 certificates are also known as trusted roots. For more information on managing an SSL key database,

This parameter effectively enables the -cZ switch.

► -cw password | ? - Use password as the password for authentication. Use the ? to generate a password prompt. Using this prompt prevents your password from being visible through the ps command.
► -cY - The password for the trusted database.

► -cZ - Use a secure SSL connection to communicate with the LDAP server. The -cZ option is only supported when the SSL component entry, as provided by IBMfs GSKit, is installed.

Example 13-3   Checking data differences between replica and master

```
ldapdiff -b <baseDN> -sh <supplierhostname> -ch <consumerhostname>
[options]
```

► This shows that there are differences between the master and replica

```
C:\>ldapdiff -b "o=ibm,c=us" -sh win2k1 -ch win2k2 -L c:\ldapdiff.ldif

Traversing the tree on both the servers...

! ou=Austin,o=ibm,c=us
  < ibm-entryuuid : 71ce21da-f737-4c38-8646-6eba6a9e03a7
  > ibm-entryuuid : c9d47e80-bb66-4aa6-8992-a374cae3a85d
  < modifyTimeStamp : 20040218032345.000000Z
  > modifyTimeStamp : 20040218035213.000000Z
  < createTimeStamp : 20040218032345.000000Z
  > createTimeStamp : 20040218035213.000000Z

! ou=In Flight Systems,ou=Austin,o=ibm,c=us
  < ibm-entryuuid : 9033545f-3675-4788-88a7-3aa77053715f
  > ibm-entryuuid : 81a86631-36e2-4f03-a55a-b8a870133c3f
  < businesscategory : aircraft
  < modifyTimeStamp : 20040218032348.000000Z
  > modifyTimeStamp : 20040218035217.000000Z
  < createTimeStamp : 20040218032348.000000Z
  > createTimeStamp : 20040218035217.000000Z

! ou=Home Entertainment,ou=Austin,o=ibm,c=us
  < ibm-entryuuid : debc5755-53f4-41b1-bbf5-27c81f0da706
  > ibm-entryuuid : d8f381c7-6c44-4dbb-8133-aa9033daf8fe
  < businesscategory : Home Entertainment
  < modifyTimeStamp : 20040218032349.000000Z
  > modifyTimeStamp : 20040218035217.000000Z
  < createTimeStamp : 20040218032349.000000Z
  > createTimeStamp : 20040218035217.000000Z

! ou=Groups,o=ibm,c=us
  < ibm-entryuuid : c5b95edf-c40f-4293-b97d-be802dd40238
  > ibm-entryuuid : bcc28748-f07e-41f6-a11d-851922d3c1bf
  < modifyTimeStamp : 20040218032349.000000Z
  > modifyTimeStamp : 20040218035217.000000Z
  < createTimeStamp : 20040218032349.000000Z
  > createTimeStamp : 20040218035217.000000Z

C:\>
```

► This shows that there are no differences between the master and replica

```
C:\>ldapdiff -b "o=ibm,c=de" -sh win2k1 -ch win2k3

Traversing the tree on both the servers...

C:\>
```

Example 13-4   Checking schema between replica and master server

```
ldapdiff -S -sh <supplierhostname> -ch <consumerhostname> [options]
    C:\>ldapdiff -S -sh win2k1 -ch win2k2
    Schema compare is in progress...
    This may take a few minutes...
    Schema compare is complete.
```

> **Note**: If no DN arguments are provided, the ldapdiff command waits to read a list of DNs from standard input. To break out of the wait, use Ctrl+C or Ctrl+D.

SSL, TLS notes for ldapdiff

To use the SSL or TLS -related functions associated with this utility, the SSL or TLS libraries and tools must be installed. The SSL or TLS libraries and tools are provided with IBMfs Global Security Kit (GSKit), which includes security software developed by RSA Security Inc.

> **Note:** For information regarding the use of 128-bit and triple DES encryption by LDAP applications, including the LDAP sample programs, see LDAP_SSL in the *IBM Directory C-Client SDK Programming Reference* (available at: http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html). This section describes the steps required to build the sample programs and your applications so they can use SSL with the strongest encryption algorithms available.

See the makefile associated with the sample programs for more information on linking an LDAP application so that it has access to 128-bit and triple-DES encryption algorithms.

The gsk7ikm utility is used to define the set of trusted certification authorities (CAs) that are to be trusted by the client. By obtaining certificates from trusted CAs, storing them in the key database file, and marking them as ftrustedf, you can establish a trust relationship with LDAP servers that use ftrustedf certificates issued by one of the trusted CAs. The gsk7ikm utility can also be used to obtain a client certificate, so that client and server authentication can be performed. If

the LDAP servers accessed by the client use server authentication only, it is sufficient to define one or more trusted root certificates in the key database file. With server authentication, the client can be assured that the target LDAP server has been issued a certificate by one of the trusted CAs. In addition, all LDAP transactions that flow over the SSL or TLS connection with the server are encrypted including the LDAP credentials that are supplied on the ldap_bind or ldap_simple_bind_s. For example, if the LDAP server is using a high-assurance VeriSign certificate, you should obtain a CA certificate from VeriSign, import it into your key database file, and mark it as trusted. If the LDAP server is using a self-signed server certificate, the administrator of the LDAP server can supply you with a copy of the serverfs certificate request file. Import the certificate request file into your key database file and mark it as trusted.

If the LDAP servers accessed by the client use client and server authentication, it is necessary to:

► Define one or more trusted root certificates in the key database file. This allows the client to be assured that the target LDAP server has been issued a certificate by one of the trusted CAs. In addition, all LDAP transactions that flow over the SSL or TLS connection with the server are encrypted, including the LDAP credentials that are supplied on the ldap_bind or ldap_simple_bind_s.

► Create a key pair using gsk7ikm and request a client certificate from a CA. After receiving the signed certificate from the CA, store the certificate in the client key database file.

LDAPDIFF diagnostics

Exit status is 0 if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error.

14

Access control

This chapter covers ITDS Access Control Lists (ACLs) and how to manage them. ACLs provide a means to protect information stored in a LDAP directory. Administrators use ACLs to restrict access to different portions of the directory, or specific directory entries. LDAP directory entries are related to each other by a hierarchical tree structure. Each directory entry (or object) contains the distinguished name of the object as well as a set of attributes and their corresponding values.

14.1 Overview

Since directories are used for storing various kinds of data, ranging from publicly accessed to highly sensitive, and are accessed by different users, therefore it is of utmost importance to restrict users from tampering with other users' data. For example, a user after logging in should not be allowed to delete or modify an entry which he did not create although he should be able to see it. This is achieved by implementing Access Control Lists (ACLs). ACLs are a means of controlling or restricting users from accessing different parts of the DIT.

The way access control is implemented in ITDS is as follows.

For each entry (In a directory server terms 'dn') that need to be access controlled, accompany it with the relevant list of the users and their corresponding permissions.

For example, if it is required to deny write access to a user "cn=user1,o=ibm,c=us" on the entry "ou=payroll,o=ibm,c=us" then we can do so by modifying the entry "ou=payroll,o=ibm,c=us". The modification required is basically the addition of a new attribute, aclEntry to "ou=payroll,o=ibm,c=us". However, if the entry "ou=payroll,o=ibm,c=us" already contains aclEntry, do not worry, this is a multivalued attribute. Just add one more value to aclEntry and we are done.

Here is what we need to add to "ou=payroll,o=ibm,c=us":

```
aclEntry=access-id:CN=USER1,O=IBM,C=US:normal:rsc:normal:deny:w
```

Do not worry too much on how exactly do we add this attribute to the desired entry or where all in the directory will that impact or what does "normal" mean, etc. We will cover this in this chapter. Currently it is essential to have just the meaning of the above line understood. The above line, when read in plain English, signifies that a user (access-id) with dn "cn=user1,o=ibm,c=us", is:

► (grant)ed (r)ead, (s)earch and (c)ompare access over the (normal) attributes.

► (deny)ed (w)rite access over the (normal) attributes.

Please read the line of aclEntry and the description following it in parallel. Reading in this manner will make the concept of ACLs very easy to understand.

We have more forms of ACL specification. The above example explicitly sets ACLs for an access-id that is, a single user. Similarly we can set the ACLs for a group and make sure that all the users with alike permissions are put into the same group. There is more to it, as will be seen, when we traverse through the rest of this chapter.

14.2 ACL model

To begin with, let us see how the ACL model looks like. The ACL model is based on two sets of attributes:

► The entryOwner information

► The Access Control Information (ACI)

In conformance with the LDAP model, the ACI and the entryOwner information both are represented as attribute-value pairs. The LDIF syntax can be used to administer these values.

14.2.1 EntryOwner information

The entry owners have complete permissions to perform any operation on the object regardless of the aclEntry. Additionally, the entry owners are the only ones who are permitted to administer the aclEntries for that object. EntryOwner is an access control subject, it can be defined as individuals, groups or roles. The attributes that define the entryOwnership are:

► entryOwner: Defines an entry owner

► ownerPropagate: Specifies whether the owner set is propagated to the children.

Note: The directory administrator and administration group members are the entryOwners for all objects in the directory by default, and this entryOwnership cannot be removed from any object.

14.2.2 Access Control information

The ACI specifies a subject's (user's) permission to perform a given operation against a LDAP object. Do not confuse this with ACL. ACL is basically a cumulative set of the entry owners and the ACI.

ACI is further split, depending upon the way intended to specify the ACLs. We can specify the ACLs, whereby we specify a set of rights to the user "cn=user1,o=ibm,c=us" over the current object. The descendants also may get impacted depending upon the setting of the aclPropagate attribute. Such ACLs are known as non-filtered ACLs. On the other hand, we can also specify the set of rights to the user "cn=user1,o=ibm,c=us" over a set of objects conforming to the filter "cn=a*", which is a more generalized way of setting ACLs. Such ACLs are called filtered ACLs. It is as easy as that. Below is the classification in more detail.

Non-filtered ACLs

This type of ACL applies explicitly to the directory entry that contains them, but may be propagated to none or all of its descendant entries. The default behavior of the non-filtered ACL is to propagate. The attributes that define non-filtered ACLs are:

- ► aclEntry - Defines a permission set

  ```
  aclentry=access-id:CN=USER1,O=IBM,C=US:normal:rsc:normal:deny:w
  ```

- ► aclPropagate - Specifies whether the permission set is propagated to the descendant entries

  ```
  aclpropagate=TRUE
  ```

Consider Figure 14-1 for a better explanation of non-filtered ACLs.
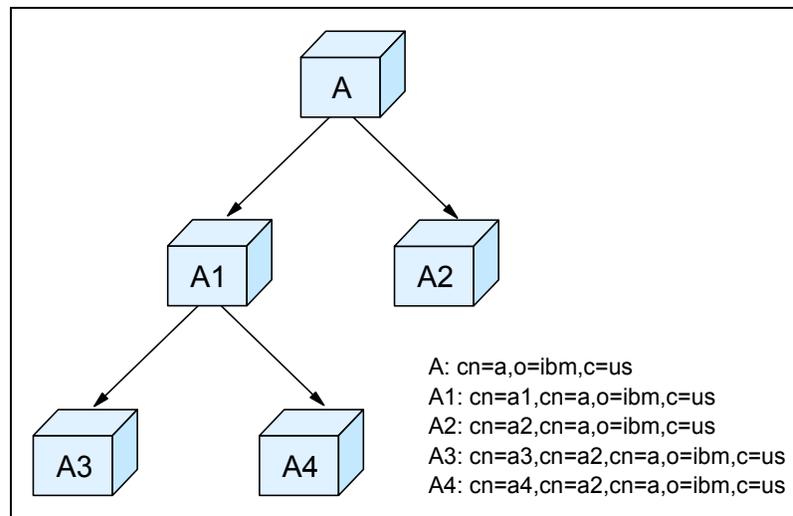
Figure 14-1   A simple Directory Information Tree (DIT)

Suppose we define the acls at entry A as:

```
aclentry=access-id:CN=USER1,O=IBM,C=US:system:deny:rsc:critical:deny:rwsc:sensi
tive:deny:rwsc:normal:rsc:normal:deny:w:object:deny:ad:restricted:deny:rwsc
```

The above aclEntry is an example of defining non-filtered acls. There are two reasons for this being an example of non-filtered acls:

- ► The attribute aclentry does not exist for non-filtered ACLs. For non-filtered ACLs we have the attribute ibm-filterAclEntry.

- ► There is no mention of the filter of the affected objects.

Consider the DIT in Figure 14-1 on page 398. Suppose we define ACLs at A, specifying "cn=user1,o=ibm,c=us" has write access over it. This is going to propagate down the tree till the leaves (If aclPropagate at A is set to true), or till the point where another set of explicitly ACLs have been defined, whichever happens to come earlier. In other words, if no other explicit ACLs have been defined at A1 and A3, both A1 and A3 will have the ACLs, as defined at A, that is, specifying a write access to "cn=user1,o=ibm,c=us" over them.

Filtered ACLs

Filter based ACLs employ a search, using a specified object filter, like "cn=user*", to select the directory entries to which they apply.

> **Note:** The key thing to remember in case of filtered ACLs is that the filter we're specifying is for the objects that will be impacted and not the subject. This filter is often misread as the set of subjects, rather than objects.

The directory entry that contains the filter ACL will serve as the base of the search. The scope of the search will be subtree, which includes the "entry containing the filter", as well as, zero, one, or more of its descendant entries.

Filter-based ACLs do not propagate in the same way that non-filter-based ACLs currently do. By nature, they inherently propagate to any comparison matched objects in the associated subtree. For this reason, the aclPropagate attribute, which is used to stop propagation of non-filter ACLs, does not apply to the new filter-based ACLs.

Consider our DIT in Figure 14-1 on page 398, if we set a filter ACL at A, with a filter (cn=a2*), then it would map to A2, and the filteracls would propagate to A2. There is not a means whereby, we can restrict ACL propagation, using attributes like aclPropagate, as like in case of non-filtered acls. This will not be the case if the ibm-filterAclInherit is set to false at A2. However, We will see that case later.

The default behavior of filter-based ACLs is to accumulate from the lowest containing entry, upward along the ancestor entry chain, to the highest containing entry in the DIT. The effective access is calculated as the union of the access rights granted, or denied, by the constituent ancestor entries. There is an exception to this behavior. For compatibility with the subtree replication feature, and to allow greater administrative control, a ceiling attribute is used as a means to stop accumulation at the entry in which it is contained. The ibm-filterAclInherit attribute is used as this ceiling attribute, which is explained later. Do not confuse this with aclPropagate. aclPropagate decides whether we can send the ACLs down the tree, whereas ibm-filterAclInherit tells where we are supposed to consider the filters defined above me, in the DIT, for access evaluation.

What this means is that if we deny write access to the filter cn=a* at the entry A to the user "cn=user1,o=ibm,c=us" and we grant access to the filter cn=a* at the entry A1, then at the time of access evaluation at A1/A3/A4 the access provided for the user at both A and A1 is taken into consideration, which happens to be an access of grant : write at A1 and an access of deny : write at A. Since deny is more stronger than grant, the effective access of user "cn=user1,o=ibm,c=us" over A1/A3/A4 turns out to be deny:write. That seems very much simple, is not it?

Filter based ACLs are maintained using the following attributes:

► ibm-filterAclEntry: It is the same form as the aclEntry attribute but has an additional component called object filter.

► ibm-filterAclInherit: When set to False, it terminates ACL accumulation. Its default value is True.

If this still seems confusing, do not worry, the following example would help.

Consider Figure 14-1 on page 398 again. Suppose we define a filter ACL entry as like below at the entry cn=a,o=ibm,c=us:

```
ibm-filterAclentry=access-id:CN=USER1,O=IBM,C=US:(cn=a*):object:deny:ad:normal:
rwsc
```

Then, here is what we are providing the user "cn=user1,o=ibm,c=us": Please note the filter first. The filter "cn=a*" means all the entries conforming to "cn=a*", which are at or below the subtree "cn=a,o=ibm,c=us", since that's where this Filter ACL has been defined. Let us consider what we have granted/denied for the user cn=user1,o=ibm,c=us. The entries conforming to cn=a* are referred to as objects for the convenience of explanation, in the next two bullets.

► user1 is (deny)ed (a)dd children under the current entry or (d)elete the current entry.

► user1 is granted (r)ead, (w)rite, (s)earch and (c)ompare access over the objects.

The objects mentioned above are the set of entries, having DNs conforming to the filter cn=a*, that is, over the users A, A1, A2, A3 and A4 mentioned above.

The above example should definitely have brought forth a clearer picture of the filtered acls. How exactly they are setup and the relevant details are the subject matter of the rest of the chapter. So do not worry, please go ahead for further details.

> **Note:** Given an entry in the DIT, we can specify either the non-filtered acls or the filtered acls over that entry. We cannot have a combination of both at the same given entry.

14.3 Access control attribute syntax

As indicated in "ACL model" on page 397, the ACL attributes can be managed using LDIF notation. The syntax of the filter ACL attributes are a minor modifications of the current non-filter based ACL attributes. The Backus Naur Form (BNF) for the ACI and entryOwner attributes is shown below:

```
<aclEntry> ::=  <subject> [ ":"  <rights> ]
<aclPropagate> ::=  "true" | "false"
<ibm-filterAclEntry> ::=  <subject>  ":" <object filter>  [ ":"  <rights> ]
<ibm-filterAclInherit> ::=  "true" | "false"
<entryOwner> ::=  <subject>
<ownerPropagate> ::= "true" | "false"
<subject> ::= <subjectDnType> ':' <subjectDn> | <pseudoDn>
<subjectDnType> ::= "role" | "group" | "access-id"
<subjectDn> ::= <DN>
<DN> ::= distinguished name as described in RFC 2251, section 4.1.3.
<pseudoDn> ::= "group:cn=anybody" | "group:cn=authenticated" |
"access-id:cn=this"
<object filter> ::= string search filter as defined in RFC 2254, section 4
(extensible matching is not supported).
<rights> ::= <accessList> [":" <rights> ]
<accessList> ::=  <objectAccess> | <attributeAccess> | <attributeClassAccess>
<objectAccess> ::= "object:" [<action> ":"]  <objectPermissions>
<action> ::= "grant" | "deny"
<objectPermisssions> ::=  <objectPermission> [ <objectPermissions> ]
<objectPermission> ::= "a" | "d" |  ""
<attributeAccess> ::= "at." <attributeName> ":" [<action> ":"]
<attributePermissions>
<attributeName> ::= attributeType name as described in RFC 2251, section 4.1.4.
(OID or alpha-numeric string with leading alphabet, "-" and ";" allowed)
<attributePermissions> ::=  <attributePermission> [<attributePermissions>]
<attributePermission> ::=  "r" | "w" | "s" | "c" |  ""
<attributeClassAccess> ::= <class> ":" [<action> ":"] <attributePermissions>
<class> ::= "normal" | "sensitive" | "critical" | "system" | "restricted"
```

Wondering what the above stuff is all about! The following lines would clarify the above contents in a more elaborate way.

14.3.1 Subject

Subject is the entry or entity which requests access to operate on a directory entry or object. It consists of a combination of DN-type and a DN. The valid DN types are access-Id, Group and Role.

For example, a subject might be "`access-id: cn=personA, o=IBM`" or "`group: cn=deptXYZ, o=IBM`". If a DN contains ":" (colon), it must be surrounded by double quotes. The double quotes which are parts a DN should be escaped with backslash character. All directory groups can be used in access control. Roles that are used in access control must have an objectclass of AccessRole.

> **Note:** Any group of the type AccessGroup, GroupOfNames, GroupofUniqueNames, or groupOfURLs structural objectclasses or the ibm-dynamicGroup, ibm-staticGroup auxiliary objectclasses can be used for access control.

Roles and Groups have similar implementations but differ conceptually. A user belonging to a role is assumed to be possessing the necessary authorities that are required to do any job associated with that Role. With group membership, there is no such associated assumptions.

Roles are similar to groups in that they are represented in the directory by an object. Additionally, roles contain a group of DNs. Roles that are used in access control must have an objectclass of AccessRole.

Let us consider an example. We have a group of users who are part of the team A. Name this group as group A. Now define a role, "player of team A". Now whoever conforms to the group A, or "player of team A" is a part of the team A. It is just a different perspective of looking at the same end-object.

14.3.2 Pseudo DNs

Pseudo DNs are maintained by the server and are used for Access Control. These pseudo DNs are used to refer to large number of DNs that possess a common characteristic in relation to either the operation being performed or the object on which the operation is being performed.

Three pseudo DNs are supported by LDAP Version 3 (just to mention that this is the version of LDAP that we are talking of and not the product itself):

► access-id: cn=this

When specified as part of an ACL, this DN refers to the bindDN, which matches the DN on which the operation is performed. For example, if an operation is performed on the object "cn=personA, ou=IBM, c=US" and the

bindDn is "cn=personA, ou=IBM, c=US", the permissions granted are a combination of those given to "cn=this" and those given to "cn=personA, ou=IBM, c=US".

Now the obvious question is where do we find the necessity of having a dn such as "cn=this" in my directory? Well, let us take an example to have this absorbed. Consider a directory for the employees of an organization as like the IBM Bluepages directory. We can find a lot of information in this directory pertaining to the employees. The user has the access to modify his/her information, as like his telephone number, contact number etc. but there are fields which can't be edited by the employees, but only an administrator should, like his manager's name. There is where we can definitely use the access-id cn=this and define the ACL to deny write on the relevant attributes.

► group: cn=anybody

When specified as part of an ACL, this DN refers to all users, even those that are unauthenticated. Users cannot be removed from this group, and this group cannot be removed from the database.

Well, this group is problematic for somebody. If we fire a simple query like, "ldapsearch -s base objectclass=*". That is, the root DSE search without any credentials, we are authenticated to the directory server with the default credentials of the group "cn=Anybody". Consequently we are able to run the rootdse search successfully. Some people might say, all the data in my directory server is confidential. We are not ready to expose any of this information. How am we supposed to hide it, because all the "people in the world" are able to see it, being the members of the group "cn=Anybody". that is, by firing anonymous searches.

> **Note:** Searches against the directory server, which are fired without any credentials are also called Anonymous searches.

Well, there a lot of decent ways of getting around this problem. The simplest way is to block the anonymous searches to the directory server. The relevant settings are to made in the directory server via webadmin, as follows:

a. Connect to the directory server via the webadmin.
b. Click **Manage connection properties**.
c. Select the **General** tab.
d. By default the Allow anonymous connections check box is ticked.
e. Uncheck this check box.

Refer to Figure 14-2 on page 404 on how to disable anonymous access to the directory.
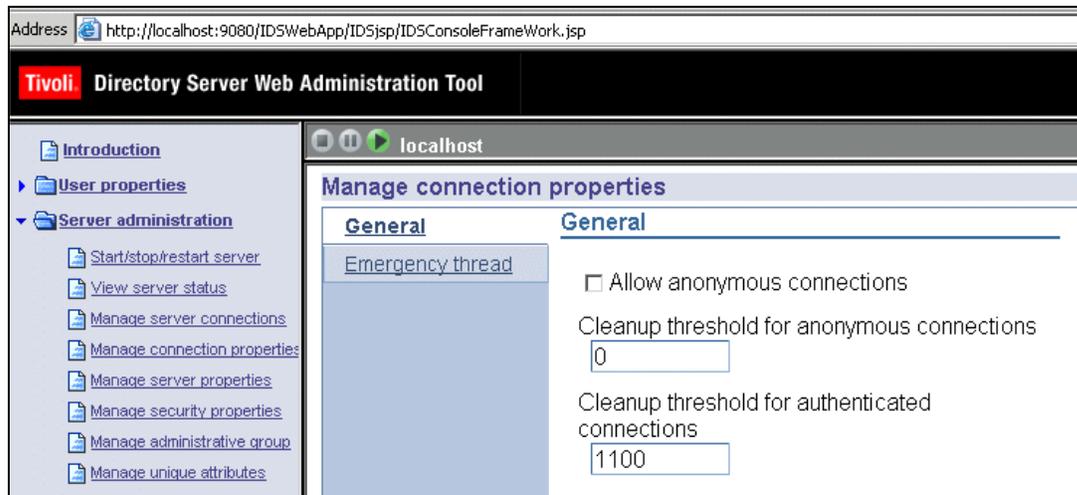
Figure 14-2   Disabling anonymous access to the directory

> **Note:** Disallowing anonymous binds might cause some applications to fail.

The other ways to get over such an issue is to make use of ACLs, wherever necessary and enforce authenticated access. We can also make use of SSL to enforce secured access to our directory. Other way is to make use of plug-ins (written by self or have them written by some service personnel, but that will cost money). There are documents, shipped along with the directory server to write plug-ins. The main purpose of the plug-in is to filter the calls coming to the directory server. That is, listen to the calls to the directory server, before the server and if no proper credentials are accompanied by the client's request, block that request and do not allow that to pass through.

The reason this group has been part of our implementation or in other words cannot be removed from the implementation is that this particular feature is part of the RFC 2251 of the LDAP protocol. Though some users do not like this feature, there are some other good reasons of keeping this group. Suppose there are a million users in our directory server. Suppose, we need to provide read access to everyone in our directory server. It is not wise enough to create a group of 1 Million members and provide a read access to this group at the relevant places. It is just a matter of tailoring the provided features to suit ones purpose.

► group: cn=Authenticated

This DN refers to any DN that has been authenticated by the directory. The method of authentication is not considered.

"cn=Authenticated" refers to a DN that has been authenticated anywhere on the server, regardless of where the object representing the DN is located. It should be used with caution, however. For example, under one suffix, "cn=Secret", there could be a node called "cn=Confidential Material", which has an aclentry of "group:cn=Authenticated:normal:rsc". Under another suffix, "cn=Common", there could be the node "cn=Public Material". If these two trees are located on the same server, a bind to "cn=Public Material" would be considered authenticated, and would get permission to the normal class on the "cn= Confidential Material" object.

14.3.3 Object filter

This parameter applies to filtered ACLs only. The string search filter as defined in RFC 2254, is used as the object filter format. Because the target object is already known, the string is not used to perform an actual search. Instead, a filter-based compare on the target object in question is performed to determine if a given set of ibm-filterAclEntry values apply to it.

Let us have this more simplified. Consider our DIT of Figure 14-1 on page 398 again. Let us suppose we define a set of ACLs for the user "cn=user1,o=ibm,c=us" at the entry A, and the filter we specify is "cn=a*". When we specify this filter, the directory server will not do an ldapsearch to find out the objects, matching the filter, and tell the user "cn=user1,o=ibm,c=us" : "Look! These are the set of objects that you have so-and-so access!" Instead, here is what the directory server would do. When we fire a query against an entry, authenticated as the user "cn=user1,o=ibm,c=us" the entry we are targeting upon, would be compared against the defined set of filters, till the top of the tree, or till the point that the filters can be chased, verify the permissions of the entry "cn=user1,o=ibm,c=us" over that entry. The filter chasing is avoided by specifying the ibm-filterAclInherit=false.

Depending upon the access rights calculated in this manner, either our intended operation succeeds or fails, for want of sufficient permissions.

14.3.4 Rights

Access rights can apply to an entire object or to attributes of the object. The LDAP access rights are discreet. One right does not imply another right. The rights may be combined together to provide the desired rights list following a set of rules discussed later. Rights can be of an unspecified value, which indicates

that no access rights are granted to the subject on the target object. The rights consist of three parts:

► Action

Defined values are grant or deny. If this field is not present, the default is set to grant.

► Permissions

There are six basic operations that may be performed on a directory object. From these operations, the base set of ACI permissions are taken. These are: Add an entry, delete an entry, read an attribute value, write an attribute value, search for an attribute, and compare an attribute value.

The possible attribute permissions are: read (r), write (w), search (s), and compare (c). Additionally, object permissions apply to the entry as a whole. These permissions are add child entries (a) and delete this entry (d).

Table 14-1 summarizes the permissions needed to perform each of the LDAP operations.

Table 14-1   Permissions needed to perform LDAP operations

Operation	Permissions needed
ldapadd	add (on parent)
ldapdelete	delete (on object)
ldapmodify	write (on attribute being modified)
ldapsearch	► search, read (on attributes in RDN) ► search (on attributes specified in the search filter) ► search (on attributes returned with just names) ► search, read (on attributes returned with values)
ldapmodrdn	write (on RDN attribute)
ldapcompare	compare (on attribute being compared)

Note: For search operations, the subject is required to have search (s) access to all the attributes in the search filter or no entries are returned. For the ldapsearch to be successful, the subject is required to have search (s) and read (r) access to all the attributes in the RDN of the entries that are expected to be returned or these entries are not returned.

► Access target

Access target refers to the scope to which the permissions apply. These permissions can be applied to the entire object (add child entry, delete entry), to an individual attribute within the entry, or can be applied to groups of attributes (Attribute Access Classes) as described below.

Attributes requiring similar access rights or permissions are grouped together in classes. Attributes are mapped to their attribute classes in the directory schema file. These classes are discrete that is, access to one class does not imply access to another class. Permissions are set with regard to the attribute access class as a whole. The permissions set on a particular attribute class apply to all the attributes within that access class, unless individual attribute access permissions are specified.

IBM defines five attribute classes that are used in evaluation of access to user attributes: Normal, sensitive, critical, system, and restricted. For example, the attribute commonName belongs to the normal class, and the attribute userPassword belongs to the critical class. All user defined attributes belong to the normal access class unless otherwise specified.

The system class attributes that apply to access control are:

– aclSource: This attribute identifies the source from which a given entry is supposed to inherit ACLs.

– ibm-effectiveAcl: This attribute gives the effective ACLs on an entry after taking into consideration all ACLs defined at self, the default ACLs and also the ACLs that are inherited.

– ownerSource: This attribute identifies the source from which a given attribute is supposed to inherit its owner.

These attributes are maintained by the LDAP server and are read-only to the directory users and administrator.

The restricted class attributes that define access control are:

– aclEntry: This attribute stores the information pertaining to non-filtered ACLs.

– aclPropagate: This attribute indicates whether the ACLs defined at this level are supposed to be propagated down the tree.

  • aclPropagate=true indicates that the acls need to be propagated, or

  • aclPropagate=false indicates that the ACL propagation stops here.

– entryOwner: This attribute stores information as to who exactly is the owner of a given entry.

– ibm-filterAclEntry: This attribute stores the information pertaining to filtered ACLs.

- ibm-filterAclInherit: This attribute indicates whether a given entry is supposed to inherit filter ACLs, from its ancestors, for evaluating effective access.

- ownerPropagate: This attribute indicates whether the owner specified at a given entry is supposed to be propagated down the tree.

  - ownerPropagate=true indicates that the owner needs to be propagated.

  - ownerPropagate=false indicates that the owner need not be propagated.

By default all users have read access to the restricted attributes but only entryOwners can create, modify, and delete these attributes.

Here is an aclEntry for a user with the permissions set on different attribute classes:

```
aclentry=access-id:CN=USER1,O=IBM,C=US:system:deny:rsc:critical:deny:rws
c:sensitive:deny:rwsc:normal:rsc:normal:deny:w:object:deny:ad:restricted
:deny:rwsc
```

The above line, when read in plain English, signifies that a user (access-id) with dn "cn=user1,o=ibm,c=us", is:

(deny)ed (r)ead, (s)earch and (c)ompare access over the (system) attributes.

> **Note:** Write to (system) attributes is denied by default for all users, including the directory administrator.

(deny)ed (r)ead, (w)rite, (s)earch and (c)ompare access over the (critical) attributes.

(deny)ed (r)ead, (w)rite, (s)earch and (c)ompare over the (sensitive) attributes.

(grant)ed (r)ead, (s)earch and (c)ompare access over the (normal) attributes.

(deny)ed (w)rite access over the normal attributes.

(deny)ed to (a)dd and (d)elete any (object) that is, children.

(deny)ed (r)ead, (w)rite, (s)earch and (c)ompare access over the (restricted) attributes.

> **Note:** denied is deliberately put up as denied, just for the sake of making co-relation between the line of the aclentry and the relevant description easier.

14.3.5 Propagation

All entries in the directory may or may not have aclEntry or entryOwner explicitly defined on them. If either of these values is not explicitly defined, it is inherited from an ancestor entry in the DIT.

Each explicit aclEntry or entryOwner applies to the entry on which it is set. Additionally, the value might apply to all descendants that do not have an explicitly set value. These values are considered propagated; their values propagate through the directory tree. Propagation of a particular value continues until another propagating value is reached.

> **Note:** Filter-based ACLs do not propagate in the same way that non-filter-based ACLs do. They propagate to any comparison matched objects in the associated subtree.

aclEntry and entryOwner can be set to apply to just a particular entry with the propagation value set to "false", or to an entry and its subtree with the propagation value set to "true". Although both aclEntry and entryOwner can propagate, their propagation is not linked in anyway.

The aclEntry and entryOwner attributes allow multiple values within the same entry, however, the propagation attributes, aclPropagate and ownerPropagate, can only have a single value within the same entry.

The system attributes aclSource and ownerSource contain the DN of the effective node from which the aclEntry or entryOwner are evaluated, respectively. If no such node exists, the value default is assigned.

Now we would consider some examples as to how the above defined attributes have been evaluated at different levels of the DIT.

Case 1: Here is the way of getting the aclSource, ownerSource, aclEntry and entryOwner for a given entry "o=ibm,c=us" via the command-line query "ldapsearch":

```
D:\>ldapsearch -s base -D <admin dn> -w <admin pw>-b o=ibm,c=us objectclass=*
aclSource ownerSource aclEntry entryOwner
o=IBM,c=US
ownerSource=default
aclSource=default
entryOwner=access-id:CN=ROOT
aclEntry=group:CN=ANYBODY:system:rsc:normal:rsc:restricted:rsc
```

"o=ibm,c=us" being the suffix entry the ownerSource and the aclSource are set to default. As seen, the entryOwner is the directory administrator (cn=root in this

case) and the ACLs are the ones set by default that is, for cn=Anybody, as none have been set explicitly.

Case 2: Now let us see the same set of attributes for which we do not specify explicit ACLs, but which inherit from o=ibm,c=us, where we have not specified any ACLs either:

```
D:\>ldapsearch -s base -D <admin dn> -w <admin pw> -b cn=user1,o=ibm,c=us
objectclass=* aclSource ownerSource aclEntry entryOwner
cn=user1,o=ibm,c=us
ownerSource=default
aclSource=default
entryOwner=access-id:CN=ROOT
aclEntry=group:CN=ANYBODY:system:rsc:normal:rsc:restricted:rsc
```

As seen above, the ownerSource and the aclSource are still set to default. Do you not think that these values have been set so, as their parent that is, o=ibm,c=us has not been set with any explicit value? There is no harm in believing so.

Time to prove it.

Case 3: If we set some ACLs at o=ibm,c=us as "aclEntry=access-id:CN=USER1,O=IBM,C=US:object:ad:normal:r", then here is what we get for the same set of attributes:

```
D:\>ldapsearch -s base -D <admin dn> -w <admin pw> -b cn=user1,o=ibm,c=us
objectclass=* aclSource ownerSource aclEntry entryOwner
cn=user1,o=ibm,c=us
ownerSource=default
aclSource=O=IBM,C=US
entryOwner=access-id:CN=ROOT
aclEntry=access-id:CN=USER1,O=IBM,C=US:object:ad:normal:r
```

Did you notice the aclSource being modified to o=ibm,c=us? That is the way it goes.

Case 4: Now let us see the same attributes for an entry which comes below "o=ibm,c=us", in the DIT. we have defined explicit ACLs at the entry which we are searching in this case:

D:\>ldapsearch -s base -D <admin dn> -w <admin pw> -b
ou=Payroll,o=ibm,c=us objectclass=* aclSource ownerSource aclEntry
entryOwner aclPropagate

ou=payroll,o=ibm,c=us
aclPropagate=TRUE
ownerSource=default
aclSource=OU=PAYROLL,O=IBM,C=US

```
entryOwner=access-id:CN=ROOT
aclEntry=access-id:CN=USER1,O=IBM,C=US:system:deny:rsc:critical:deny:rwsc:sensitiv
e:deny:rwsc:normal:rsc:normal:deny:w:object:deny:ad:restricted:deny:rwsc
```

As seen above, the ownerSource is still default. Since ACLs have been explicitly defined at this entry, the aclSource happens to be the same entry. we have shown here one more flag aclPropagate, for a specific reason, which would be clearer down the line.

Case 5: Now let us see the same set of attributes for an entry which is actually inheriting ACLs from an entry, where ACLs were explicitly defined. The same attributes for an entry below ou=Payroll,o=ibm,c=us would be seen as:

```
D:\>ldapsearch -s base -D <admin dn> -w <admin pw> -b
cn=accountant,ou=Payroll,o=ibm,c=us objectclass=* aclSource ownerSource
aclEntry entryOwner
cn=accountant,ou=payroll,o=ibm,c=us
ownerSource=default
aclSource=OU=PAYROLL,O=IBM,C=US
entryOwner=access-id:CN=ROOT
aclEntry=access-id:CN=USER1,O=IBM,C=US:system:deny:rsc:critical:deny:rwsc:sensi
tive:deny:rwsc:normal:rsc:normal:deny:w:object:deny:ad:restricted:deny:rwsc
```

As seen above the aclSource happens to be "OU=PAYROLL,O=IBM,C=US", which happens to be its parent entry and we do not have o=ibm,c=us anywhere in the picture. The reason for this, of course you might have guessed by now, that the aclPropagate at ou=Payroll,o=ibm,c=us was set to true. Curious to know what happens if we set the same to false? Check out the next case.

Case 6: Let us see the result in that case:

```
D:\>ldapsearch -s base -D <admin dn> -w <admin pw> -b
cn=accountant,ou=Payroll,o=ibm,c=us objectclass=* aclSource ownerSource
aclEntry entryOwner

cn=accountant,ou=payroll,o=ibm,c=us
ownerSource=default
aclSource=O=IBM,C=US
entryOwner=access-id:CN=ROOT
aclEntry=access-id:CN=USER1,O=IBM,C=US:object:ad:normal:r
```

Is that not what you were expecting? The ACLs from o=ibm,c=us are inherited in this case, rather than ou=Payroll,o=ibm,c=us.

Let us sum this up. An object's effective access control definitions can be derived by the following logic:

► If there is a set of explicit access control attributes at the object, then that is the object's access control definition.

- If there is no explicitly defined access control attributes, then traverse the directory tree upwards until an ancestor node is reached with a set of propagating access control attributes.

- If no such ancestor node is found, the default access, as described in the following section, is granted to the subject.

> **Note:** The attributes pertaining to ACLs, that is, aclEntry, entryOwner, aclPropagate, ownerPropagate, aclSource, ownerSource, ibm-filterAclEntry and ibm-filterAclInherit are operational attributes.In the sense that they do not get dumped when we run the db2ldif or the ldapsearch tool against the server. An explicit mention of these attributes is required while firing ldapsearch, for these to get dumped. Examples of how to get up to the ACL attributes has already been explained in the CASEs mentioned lately.

14.3.6 Access evaluation

Access for a particular operation is granted or denied based on the subject's bind DN for that operation on the target object. Processing stops as soon as access can be determined.

The checks for access are done by first determining the entry ownership and then evaluating the object's ACI values.

Filter-based ACLs accumulate from the lowest containing entry, upward along the ancestor entry chain, to the highest containing entry in the DIT. The effective access is calculated as the union of the access rights granted, or denied, by the constituent ancestor entries. The existing set of specificity and combinatory rules are used to evaluate effective access for filter based ACLs.

Filter-based and non-filter-based attributes are mutually exclusive within a single containing directory entry. Placing both types of attributes into the same entry is not allowed, and is a constraint violation. Operations associated with the creation of, or updates to, a directory entry fail if this condition is detected.

When calculating effective access, the first ACL type to be detected in the ancestor chain of the target object entry sets the mode of calculation. In filter-based mode, non-filter-based ACLs are ignored in effective access calculation. Likewise, in non-filter-based mode, filter-based ACLs are ignored in effective access calculation.

To limit the accumulation of filter-based ACLs in the calculation of effective access, an ibm-filterAclInherit attribute set to a value of "false" may be placed in any entry between the highest and lowest occurrence of ibm-filterAclEntry in a

given subtree. This causes the subset of ibm-filterAclEntry attributes above it in the target object's ancestor chain to be ignored. The resulting access resolves to the default filter ACL value.

By default, the directory administrator, administration group members, and the master server (or peer server for replication, that is, ibm-slapdMasterDN) get full access rights to all objects in the directory except write access to system attributes. Other **e**ntryOwners get full access rights to the objects under their ownership except write access to system attributes. By default all users have read access rights to normal, system, and restricted attributes. If the requesting subject has entryOwnership, access is determined by the above default settings and access processing stops.

If the requesting subject is not an entryOwner, then the ACI values for the object entries are checked. The access rights as defined in the ACLs for the target object are calculated by the specificity and combinatory rules.

Specificity rule

The most specific aclEntry definitions are the ones used in the evaluation of permissions granted/denied to a user. The levels of specificity are:

► Access-id is more specific than group or role. Groups and roles are on the same level.

► Within the same dnType level, individual attribute level permissions are more specific than attribute class level permissions.

► Within the same attribute or attribute class level, deny is more specific than grant.

For example, if a defined ACI entry contains an access-id subject DN that matches the bind DN, then the permissions are first evaluated based on that aclEntry. Under the same subject DN, if matching attribute level permissions are defined, they supersede any permissions defined under the attribute classes. Under the same attribute or attribute class level definition, if conflicting permissions are present, denied permissions override granted permissions.

Let us take some examples to absorb this.

Consider our DIT as in Figure 14-1 on page 398. Suppose we create a group and name it as "Group1". We add "cn=user1,o=ibm,c=us" to "Group1". Now at the entry A, we are setting two sets of ACLs: We are providing "rsc", that is, (r)ead, (s)earch and (c)ompare access to "Group1" and denying write to it. Now when we bind as the user "cn=user1,o=ibm,c=us", we are denied write on A, as the group to which we belong, is denied for writes. Now, we set the ACLs for user "cn=user1,o=ibm,c=us", whereby we are giving write access to this user. Now, when we bind as "cn=user1,o=ibm,c=us", what should we be allowed to do? We

will be allowed to write, as the access-id is more specific than group. That clarifies point 1 specified above.

Now let us go to the next point. Suppose, in the entry A, we provide "rsc" access to "cn=user1,o=ibm,c=us" over the "normal attributes". We provide the "rwsc" access to the same user over the attribute "telephoneNumber" for this entry. Now what should we be allowed to do with the attribute "telephoneNumber", when we bind as the user "cn=user1,o=ibm,c=us"? Isn't that obvious that we are given write access, though the corresponding attribute class (normal) is denied of the write? The reason being, of course that the attribute telephoneNumber is explicitly allowed for writes.

That should clarify point 2 mentioned above.

Now, let us go to the next point. Suppose we set the ACLs at entry A, with aclPropagate set to true. We set the ACLs whereby, we deny write to the attribute "telephoneNumber" in the entry A. That propagates down the tree and appears in A1, assuming that we have not specified ACLs explicitly at A1. Now even if we give a "rwsc" access to the user "cn=user1,o=ibm,c=us" over the attribute "telephoneNumber" in the entry A1, the user will not be allowed to write to that attribute, because the deny which propagated from the parent is more specific than grant. Hope this clears all the three rules of specificity.

Combinatory rule

Permissions granted to subjects of equal specificity are combined. If the access cannot be determined within the same specificity level, the access definitions of lesser specific level are used. If the access is not determined after all defined ACIs are applied, the access is denied.

For example, consider the two cases of ACIs defined on cn=user1,o=ibm,c=us described below:

► Case 1:
  – access-id: cn=this: at.attribute1:grant:rws
  – access-id: cn=user1,o=ibm,c=us:at.attribute1:grant:rs:at.attribute1:deny:w

  In the above case, the (w)rite permission on attribute1 will be denied to the user cn=user1,o=ibm,c=us as access cannot be explicitly determined.

► Case 2:
  `(cn=user1,o=ibm,c=us belongs to group cn=group1)`
  – access-id: cn=this: at.attribute1:grant:rws
  – access-id: cn=user1,o=ibm,c=us:at.attribute1:grant:rs:at.attribute1:deny:w
  – group:cn=group1:at.attribute1:grant:w

In this case, after failing to determine access at the specificity level of access-id, the access definitions of lesser specific levels (group) is determined. Since, the group has (w)rite permissions on attribute1, write permission will be granted to cn=user1,o=ibm,c=us.

That was simple stuff, we believe.

> **Note:** After a matching access-id level aclEntry is found in access evaluation, the group level aclEntries are not included in access calculation. The exception is that if the matching access-id level aclEntries are all defined under cn=this, then all matching group level aclEntries are also combined in the evaluation.
>
> A defined null value permission prevents the inclusion of less specific permission definitions.
>
> Group and Role membership is determined at bind time and last until either another bind takes place, or until an unbind request is received. Nested groups and roles, that is a group or role defined as a member of another group or role, are not resolved in membership determination nor in access evaluation.

14.3.7 Working with ACLs

In this section we discuss working with ACLs.

Using the Web Administration Tool

This is to view ACL properties using the Web Administration Tool utility and to work with ACLs.

Select a directory entry. For example, ou=Payroll,o=ibm,c=US.

Click **Edit ACL**. The relevant panel shows up with the Effective ACLs tab preselected.

Refer to Figure 14-3 on page 416 on how to edit an ACL.
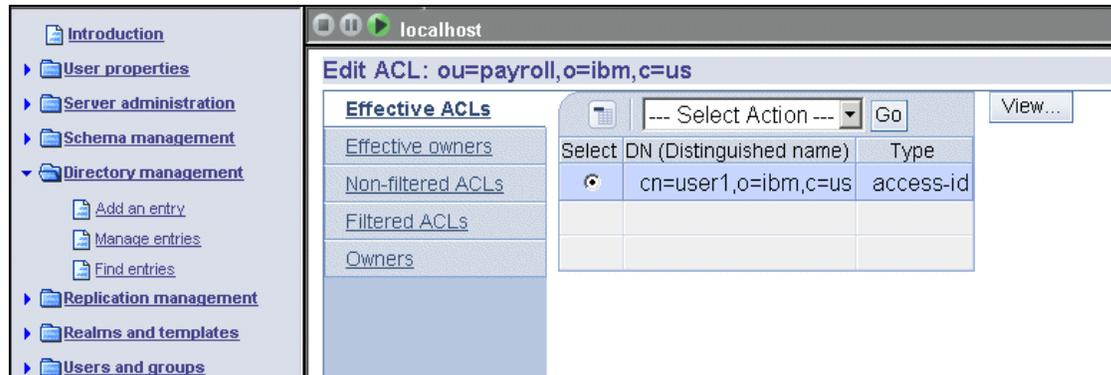
Figure 14-3   Edit ACL

This panel has five tabs:

► Effective ACLs
► Effective owners
► Non-filtered ACLs
► Filtered ACLs
► Owners

The Effective ACLs and Effective owners tabs contain read-only information about the ACLs.
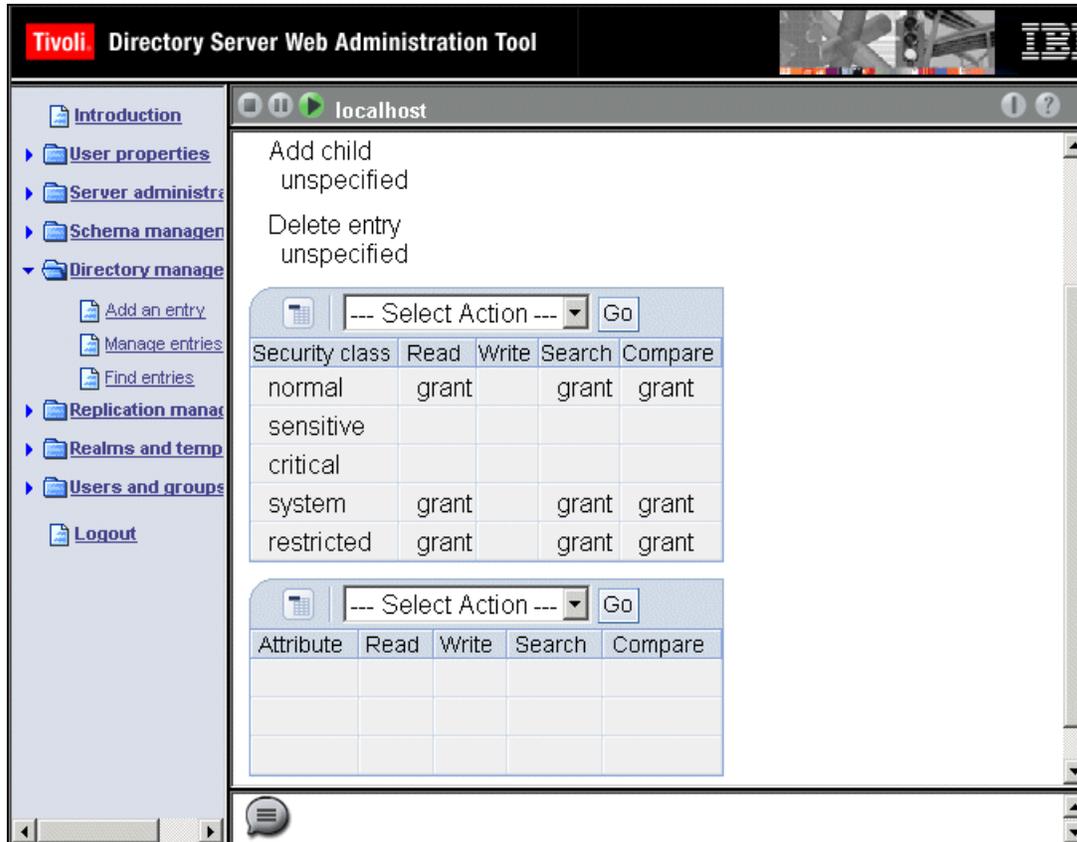
Figure 14-4   Effective ACLs

Effective ACLs

Effective ACLs are the explicit and inherited ACLs of the selected entry. We can view the access rights for a specific effective ACL by selecting it and clicking the **View** button. The panel (View access rights panel) in which we are supposed to click the View button is put up in Figure 14-3 on page 416. The following three sections appear, as shown in Figure 14-4.

► The Rights section:

 – *Add child* grants or denies the subject the right to add a directory entry beneath the selected entry.

 – *Delete entry* grants or denies the subject the right to delete the selected entry. In the above example, the Add child and the Delete entry are left unspecified, which is taken as a "Deny". Now the obvious query would be that if already have "Deny" in place, then why do we need to make use of "Unspecified". Well, reason that the "Unspecified" is kept is that it is an

indication that there would be no relevant definitions at this level of the DIT. The relevant values would be propagated by whatever gets propagated down the tree. In this case, there wasn't as yet anything to propagate down the tree, hence this was left as Unspecified.

► The Security class section defines permissions for security classes. Attributes are grouped into a set of classes, known as the security classes, depending upon the amount of security associated with them. Here are the list of possible security classes, an attribute may fall into:

– Normal - Normal attributes are the ones requiring the least security, for example, the attribute commonName or cn.

– Sensitive - Sensitive attributes are the ones requiring a moderate amount of security, for example homePhone.

– Critical - Critical attributes are the ones requiring the most security, for example, the attribute userpassword.

– System - System attributes are read only attributes that are maintained by the server.

– Restricted - Restricted attributes are the ones, used to define access control.

Each security class has one or more of the following permissions associated with it:

– Read - The subject can read attributes.
– Write - The subject can modify the attributes.
– Search - The subject can search attributes.
– Compare - The subject can compare attributes.

Click **OK** to return to the Effective ACLs tab. Click **Cancel** to return to the Edit ACLs tab.

Effective owners

Effective owners are the explicit and inherited owners of the selected entry.

Refer to Figure 14-5 on page 419 for how this section appears in the Web Administration Tool.
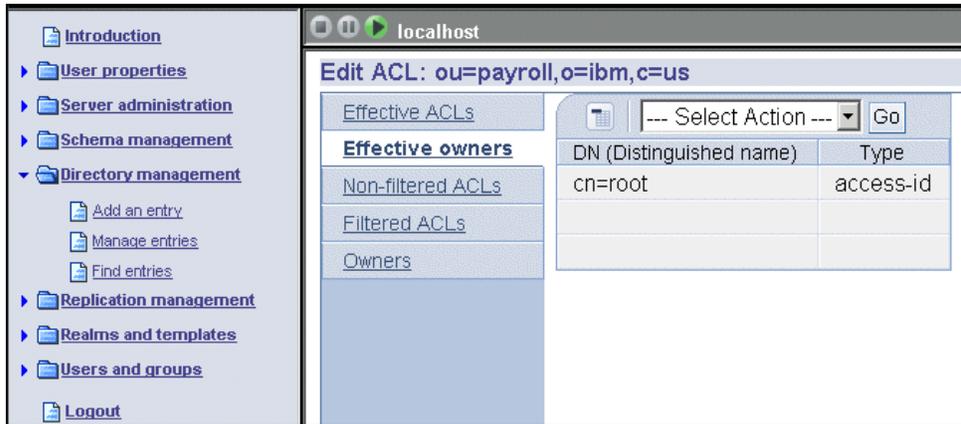
Figure 14-5   Effective owners

Non-filtered ACLs

We can use this tab for adding new non-filtered ACL entries or modifying existing non-filtered ACL informations. When we click the **Non-filtered ACLs** link, we get a panel as shown in Figure 14-6.
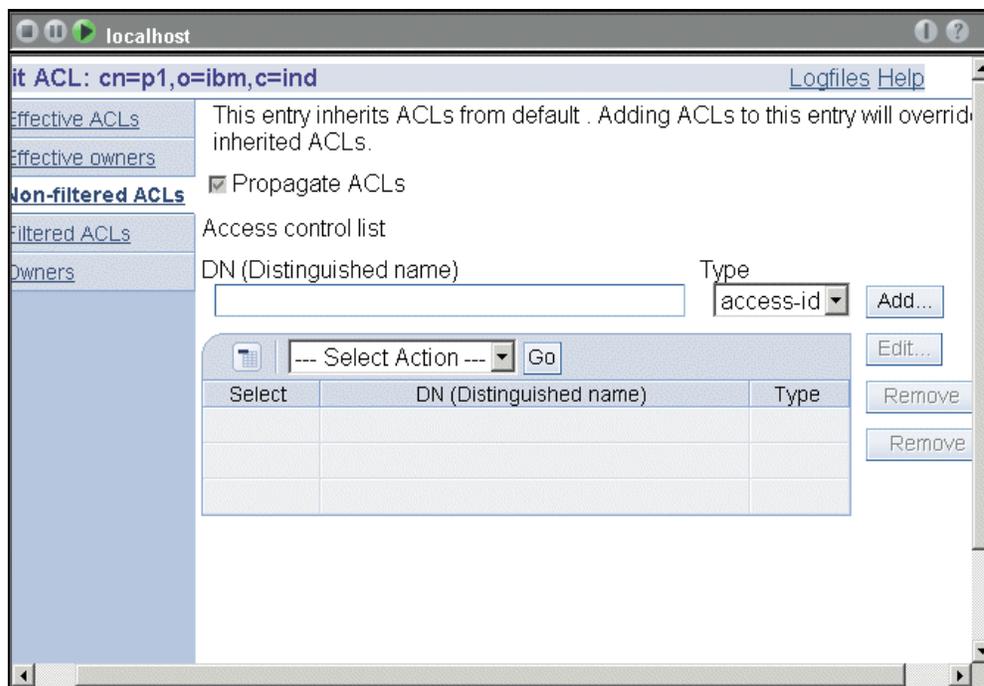
Figure 14-6   Non-filtered ACLs

Supply the necessary information above, taking into consideration the following:

► Propagate ACLs - Select the Propagate check box to allow descendants without an explicitly defined ACL to inherit from this entry. If the check box is selected, the descendent inherits ACLs from this entry and if the ACL is explicitly defined for the child entry, then the ACL which was inherited from parent is replaced with the new ACL that was added. If the check box is not selected, descendant entries without an explicitly defined ACL will inherit ACLs from a parent of this entry that has this option enabled. This point is already explained in our Case Studies earlier.

► DN (Distinguished Name) - Enter the Distinguished name of the entity requesting access to perform operations on the selected entry, for example, cn=Marketing Group.

► Type - Enter the Type of DN. For example, select access-id if the DN is a user.

Adding and editing access rights

There are two ways of setting the access rights on an object:

► Click the **Add** button to add the current/new subject DN in the list of subject DNs (Distinguished Name). (or)

► Select a dn from the existing list of subject dns and click the **Edit** button to modify the ACLs pertaining to the selected DN.

The Add access rights and Edit access rights panels, which appear after clicking Add or Edit, allows us to set the access rights for a new or existing Access Control List (ACLs). The Type field defaults to the type we selected on the Edit ACL panel. If we are adding an ACL, all other fields default to blank. If we are editing an ACL, the fields contain the values set last time the ACL was modified.

To set access rights in the Rights section:

► Grant/Deny permissions to add a child.

► Grant/Deny permissions to delete the entry itself.

► Grant/Deny Read, Write, Search and Compare permissions to different security classes of attributes.

► Define an attribute and explicitly Grant/Deny Read, Write, Search and Compare permissions to it. These permissions are more specific than the permissions on the attribute classes.

Refer to Figure 14-7 on page 421 for the panel where we are supposed to specify new/modify existing Access rights.
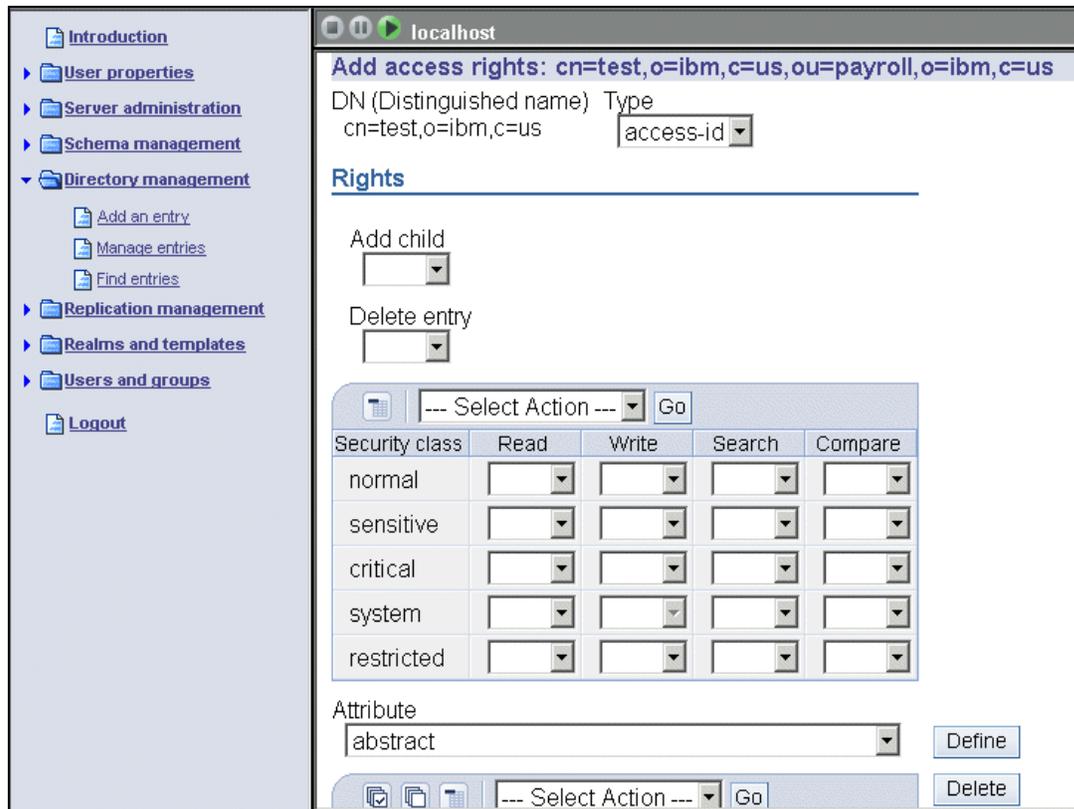
Figure 14-7   New ACLs specified

The panel to Edit ACLs is as like the above, the only difference being the title of "Edit Access Rights : <subject dn>**".**

Removing ACLs

We can remove ACLs in either of two ways:

► Select the radio button next to the ACL we want to delete. Click **Remove**.

► Click **Remove all** to delete all DNs from the list.

Figure 14-8 on page 422 shows these buttons.

Filtered ACLs

This tab can be used for adding new filtered ACLs or editing existing filtered ACLs. When we click the Filtered ACLs tab, we get the following screen.
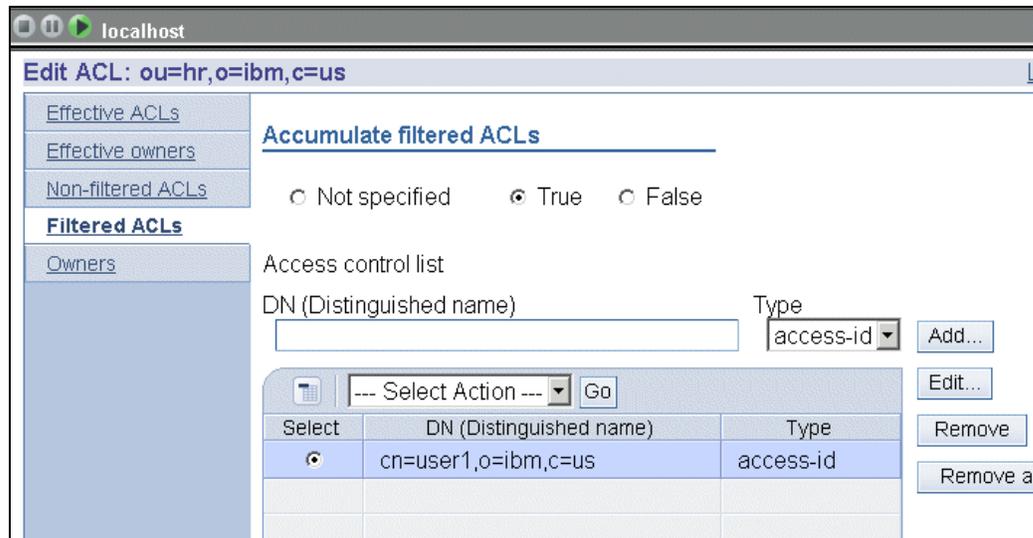
Figure 14-8   Filtered ACLs

We need to fill the following fields:

▶ Accumulate filtered ACLs

 – Select the **Not specified** radio button to remove the `ibm-filterACLInherit` attribute from the selected entry.

 – Select the **True** radio button to allow the ACLs for the selected entry to accumulate from that entry, upward along the ancestor entry chain, to the highest filter ACL containing entry in the DIT.

 – Select the **False** radio button to stop the accumulation of filter ACLs at the selected entry.

▶ DN (Distinguished Name) - Enter the (DN) Distinguished name of the entity requesting access to perform operations on the selected entry, for example, cn=manager,ou=hr,o=ibm,c=us.

▶ Type - Enter the Type of DN. For example, select access-id if the DN is a user.

Once the above fields have been entered, we need to click the appropriate button that is, either Add, Edit, Remove or Remove All, depending upon what operation we want to do.

Adding and editing access rights

Click the either the Add button to add the DN in the DN (Distinguished Name) field to the ACL list or the Edit button to modify the ACLs of an existing DN.

Refer to Figure 14-9 for the add/edit access rights.

Figure 14-9   Add Filter ACLs

To set access rights:

1. In the Rights section:

    a. Grant/Deny permissions to add a child.

    b. Grant/Deny permissions to delete the entry itself.

2. In the Filter section, enter an object filter like objectclass=person, depending upon to which all descendant objects in the DIT this ACLs should apply. The current filtered ACL propagates to any descendant object in the associated subtree that matches the filter in this field. We have already dealt with filters in one of our earlier sections. Hope you remember the concept of filters! Feel free to go back and browse through the concept again, if needed.

3. Grant/Deny Read, Write, Search and Compare permissions to different attribute classes (security class).

4. Define an attribute and explicitly Grant/Deny Read, Write, Search and Compare permissions to it. These permissions are more specific than the permissions on the attribute classes.

Removing ACLs

We can remove ACLs in either of two ways:

1. Select the radio button next to the ACL, that you want to delete. Click **Remove**.

2. Click **Remove all** to delete all DNs from the list.

This is very much like the case of non-filtered ACLs.

Providing access on the attributes

This part is common to both the filtered and non-filtered ACLs. Suppose we want to specify the ACLs over the individual attributes here is how we do: Once the tab Add Access Rights comes up, where we specify the ACLs for the attribute classes or the permissions to add children, or delete entries, there is a section at the bottom with the heading Attribute, below which there is a drop-down of attributes. We select the attribute to be access controlled and click **Define**. We would get the selected attribute added on the panel, next to which there would be dropdowns for specifying the access-rights. If we do not specify any or if we click **Cancel** the attribute would go back to the dropdown and will not appear on the panel, else (that is, when we click **OK**) it would appear on the Panel. Figure 14-10 shows the attribute, once it is defined.
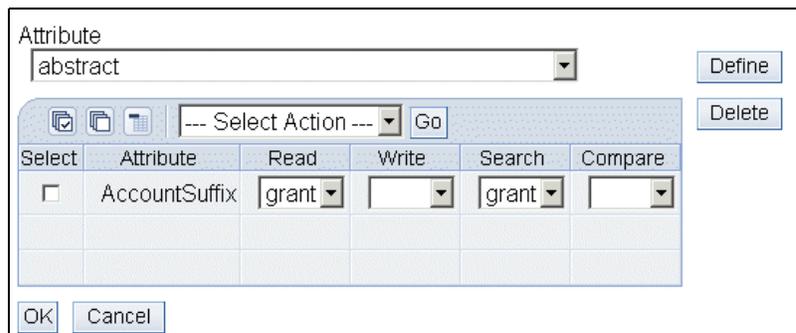
Figure 14-10   Portion of the panel for making attributes access controlled

If we need to delete the attribute, just select the attribute, using the check-box at the left of the attribute and click **Delete**.

Owners

Entry owners can be explicit or propagated (inherited).

Enter the following information on the Owners tab:

► Select the **Propagate owners** check box to allow descendants without an explicitly defined owner to inherit from this entry. If the check box is not selected, descendant entries without an explicitly defined owner will inherit owner from a parent of this entry that has this option enabled.

► DN (Distinguished Name) - Enter the Distinguished Name of the entity requesting access to perform operations on the selected entry, for example, cn=Marketing Group.

► Type - Enter the Type of DN. For example, select access-id if the DN is a user.

Adding an owner

Click **Add** to add the DN (specified in the DN(Distinguished name) field) to the list of already existing Owners or it can be the first entry to click the list.

Figure 14-11 shows the panel for an entry, for which we have explicitly specified an owner, "cn=manager,o=ibm,c=us":
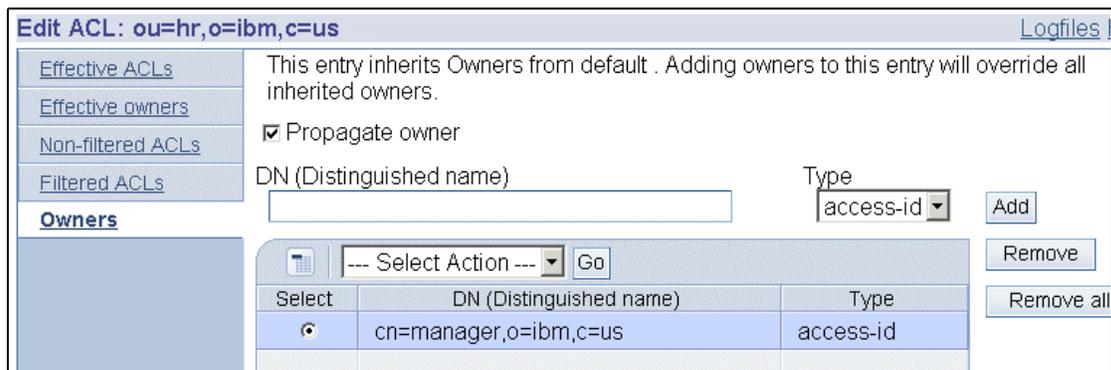
Figure 14-11   Owners of an entry

Removing an owner

We can remove an owner in either of two ways:

► Select the radio button next to the owner's DN that we want to delete. Click **Remove**.

► Click **Remove all** to delete all owner DNs from the list.

That was all with the activities pertaining to ACLs that can be performed via the GUI. Let us see how the similar activities can be done via command line.

Using command line utilities to manage ACLs

The following sections provide information on how to use command line utilities to manage ACLs.

Adding ACIs and entry owners

The following example shows how to add an entryOwner(cn=owner,o=ibm,c=us) for a given entry (cn=person1,o=ibm,c=us). Create an ldif file say acl.ldif, with the following contents:

```
dn: cn=person1,o=ibm,c=us
objectclass: person
cn: person1
sn: person1
entryowner: access-id:cn=owner,o=ibm,c=us
ownerPropagate: True
```

Add the above LDIF using the following syntax:

```
# ldapadd -D <admin dn> -w <admin password> -f acl.ldif
```

In a similar manner, we can add a group or role as an entry owner. The above example was for an (access-id) as the entry owner. The other examples shown below, under the section of "Adding ACLs and Entry Owners" should follow similar method for the additions.

The next example shows how an access ID "cn=Person 1, o=IBM,c=US" is being given permissions to read, search, and compare the (at)tribute attribute1. The permissions apply to any node in the entire subtree, at or below the node containing this ACI, that matches the "(objectclass=groupOfNames)" comparison filter. The accumulation of matching ibm-filteraclentry attributes in any ancestor nodes has been terminated at this entry by using our ceiling attribute. That attribute is the ibm-filterAclInherit attribute. It is been set to "false".

```
dn: cn=person1,o=ibm,c=us
objectclass: person
cn: person1
sn: person1
ibm-filterAclEntry:
access-id:cn=Person1,o=IBM,c=US:(objectclass=groupOfNames):at.attribute1:grant:
rsc
ibm-filterAclInherit: false
```

The next example shows how a role "cn=System Admins,o=IBM,c=US" is being given permissions to (a)dd objects below the node o=ibm,c=us, and (r)ead, (s)earch and (c)ompare (at)tribute attribute2 and the (critical) attribute class. The permission applies only to the node containing this ACI. This is achieved by setting the aclPropagate attribute to false.

```
dn: o=ibm,c=us
```

```
objeclass: organization
o: ibm
aclEntry: role:cn=System
Admins,o=IBM:object:grant:a:at.attribute2:grant:rsc:critical:grant:rsc
aclPropagate: false
```

Modifying ACI and entryOwner values

Like other attributes, the ACL attributes (except the system attributes) can be modified using ldapmodify and follow the general syntax as shown below:

```
dn: some entry
changetype: modify
<action>: <acl-attribute>
<acl-attribute>: <value>
```

Where:

► *action* (without the "<" braces) is one of the following:

  – *replace*: If the attribute value does not exist, create the value. If the attribute value exists, replace the value.

  – *add*: If the ACI or entryOwner does not exist, the ACI or entryOwner with the specific values is created. If the ACI or entryOwner exists, then add the specified values to the given ACI or entryOwner.

  – *delete*: Deletes an ACL entry with a given value.

► *acl-attribute* is one of entryOwner, ownerPropagate, aclEntry, aclPropagate, ibm-filterAclEntry, or ibm-filterAclInherit.

► *value* is the value of the given attribute.

For example, consider any entry cn=person1,o=ibm,c=us with the following acl definition:

1) `aclentry=access-id:CN=ABC:object:deny:d:object:a`

2) `aclentry=access-id:CN=P1,O=IBM,C=US:normal:rwsc:object:a`

In order to remove the acl entry cn=ABC, the syntax of ldapmodify will be:

```
ldapmodify -D <admindn> -w <adminpw>
dn: ou=person1,o=ibm,c=us
changetype: modify
delete: aclentry
aclentry: access-id:CN=ABC:object:deny:d:object:a
```

> **Note:** All the four lines after the line of ldapmodify, depicted above, can be put into an ldif file and that can be passed over to ldapmodify using the -f option. After all, ldapmodify and ldapadd ultimately boil down to the same utility.

After the above command, only the second (2) aclentry
"aclentry=access-id:CN=P1,O=IBM,C=US:normal:rwsc:object:a" remains.

If in the above ldapmodify operation, the value of acl entry to be removed is given as:

```
ldapmodify -D <admindn> -w <adminpw>
dn: ou=person1,o=ibm,c=us
changetype: modify
delete: aclentry
aclentry: access-id:CN=ABC:object:deny:d
```

> **Note:** We have not given the object add (object:a) permission in the aclentry value.

In such a scenario, both the acl entries will remain but the deny permissions on object delete (object:deny:d) will be removed from the first acl entry, this is, the value of "Delete entry" in the acl entry will be changed to "unspecified" form "deny".

Searching ACI and entryOwner values

Suppose we have an entry ou=payroll,o=ibm,c=us and we want to see all the information pertaining to acls for that entry. Here is how that can be done:

```
E:\>ldapsearch -D <admin dn> -w <admin pw> -b ou=payroll,o=ibm,c=us
objectclass=* aclEntry aclPropagate entryOwner ibm-filterAclEntry
ibm-filterAclInherit ownerPropagate

ou=payroll,o=ibm,c=us
ownerPropagate=TRUE
aclPropagate=FALSE
entryOwner=access-id:CN=ROOT
aclEntry=access-id:CN=USER1,O=IBM,C=US:system:deny:rsc:critical:deny:rwsc:sensi
tive:deny:rwsc:normal:rwsc:restricted:deny:rwsc

cn=accountant,ou=payroll,o=ibm,c=us
ownerPropagate=TRUE
aclPropagate=TRUE
entryOwner=access-id:CN=ROOT
aclEntry=access-id:CN=USER1,O=IBM,C=US:object:ad:normal:r
```

Two entries are returned as shown above, with the ACL showing that these are non-filtered ACLs.

Let us see the same search run against an entry with filtered-acls in it:

```
E:\>ldapsearch -D cn=root -w root -b ou=hr,o=ibm,c=us objectclass=* aclEntry
aclPropagate entryOwner ibm-filterAclEntry ibm-filterAclInherit ownerPropagate
```

```
ou=hr,o=ibm,c=us
ownerPropagate=TRUE
ibm-filterAclInherit=TRUE
entryOwner=access-id:CN=ROOT
ibm-filterAclEntry=access-id:CN=USER1,O=IBM,C=US:(uid=*):object:deny:ad:normal:
rwsc
```

Now let us sum up what we have learned in this chapter.

14.4 Summary

The following presents a summary from this chapter:

► ACLs are a means of protecting our information from unauthenticated access.

► ACLs are a means of providing different users, a different abstraction of the data contained in the repository, base on their roles or need to know.

► The ACL model encompasses two main parts:

   – EntryOwner information: Information pertaining to who owns the entry.

   – ACI or the Access Control Information: This is the main ingredient of the ACL model, describing the individual or group-wise access rights.

► Then we saw the classification of ACLs into the following:

   – Non-filtered ACLs: These are the ACLs where we specify the subject and the object clearly. The object that is going to get impacted is the entry where the ACLs are defined and the descendants, provided the aclPropagate flag is set to true.

   – Filtered ACLs: These are the ACLs where we specify the impacted objects, by means of a filter. Hence this is a more generalized specification of ACLs.

► Then we saw the BNF of the Access Control Information and the detailed explanation of the same.

► Then we saw how exactly the ACL Propagation takes place.

► Thereafter, we saw how exactly the ACLs get evaluated. Under this we saw two rules of ACL evaluation:

   – The specificity rules
   – The combinatory rules

► Thereafter we saw the different ways of working with the ACLs, as like:

   – The WebAdmin way for the people fond of GUI

– The command line way for the ones who love to run scripts more than pressing buttons

15

Securing the directory

This chapter mainly deals with making your directory server secure at different levels from low to high, depending upon the requirements. It describes various security features provided by the IBM Tivoli Directory Server. It describes configuring the directory for using those security features. A brief description of certificate management using the gsk7ikm key management utility is also provided.

15.1 Directory security

Security is very important in the networked world of computers, and that is equally true for directories as well. Directories are likely to contain sensitive information that needs to be protected from unauthorized access and modification. When sending data over the wire, internally or externally, sensitive information may also need to be protected against eavesdropping and modification during transportation. There is a need to know who is requesting the information and who is sending it.

The IBM Tivoli Directory Server has the following built-in features for enhanced security:

► Authentication: Ensures that the user is who he/she claims to be. It is implemented using SASL/CRAM-MD5 mechanism and certificates using SASL/SSL.

► Password Policy Enforcement: Set of rules that controls how passwords are used and administered.

► Password Encryption: Protects passwords stored in the directory from unauthorized access by encrypting it using different encryption mechanisms.

► SSL/TLS Support: Ensures tamper proof data transfer over the network.

► Protection against DOS attacks: Ensures the Directory Server remains functional under deliberate or unintended massive client requests, such as Denial of Service Attacks.

► Access Control: Ensures that the users have proper access to directory objects, before returning the required information and hence provides confidentiality.

Now let us see the above points in greater detail.

15.2 Authentication

In an LDAPv3 implementation, the client must authenticate itself to the directory service before accessing any data in the directory, otherwise access is denied. The access denial is mainly achieved by either not providing the client what it requires or by throwing an appropriate error to the client.IBM Tivoli Directory Server supports the following types of authentications that are presented next.

15.2.1 Anonymous authentication

Anonymous authentication is useful for read-only access of directory data where that data is not sensitive, such as peoples' e-mail addresses or office numbers. Essentially, that data can be made accessible to anyone. To request anonymous authentication, simple authentication is performed, against the directory server, with a distinguished name (DN) that is empty.

For example, to do an ldapsearch with anonymous binding, that is, do not include the -D (bind DN) and -w (password) options.

```
ldapsearch -b <basedn> -s <scope> <filter>
```

> **Note:** There is no way to log into the Web Administration Tool anonymously. You have to provide a user name and a password that exists in the directory or you can log in as the directory administrator.

Here is an instance of the root DSE search with an anonymous authentication:

```
C:\>ldapsearch -s base objectclass=* | grep -i config
ibm-slapdisconfigurationmode=FALSE
```

Now let us try fetching some data from the LDAP server which requires access:

```
C:\>ldapsearch -s base -D <adminDN> -w <adminPW> -b o=ibm,c=us objectclass=*
aclentry
    o=IBM,c=US
    aclentry=access-id:CN=USER1,O=IBM,C=US:object:ad:normal:r
C:\>ldapsearch -s base -b o=ibm,c=us objectclass=* aclentry
C:\>
```

You must have noticed the difference; if you do not have the proper access, the data will not be shown to you.

15.2.2 Basic authentication

Basic authentication provides authentication facilities with the DN and password transmitted over the network in clear text. Use of clear text passwords is not recommended over open networks when there is no authentication or encryption being performed by a lower layer, such as SSL (described in one of the forthcoming sections). Access (read or write) to directory data is granted based on DNs contained in the access control list of the object and/or attributes in the access request. The following is an example of searching the directory using basic authentication:

```
ldapsearch -D cn=root -w root -b <basedn> -s <scope> <filter>
```

In the above query we are assuming that the directory admin DN would be "cn=root" and the admin Password would be "root".

The above search was with regards the admin DN as the bind DN to the directory server. It is equally correct to have a bind DN which is not the admin DN but which has the necessary ACLs to access the desired data. Here is an example of authenticated access, using a user "cn=user1,o=ibm,c=us":

```
ldapsearch -D "cn=user1,o=ibm,c=us" -w user1 -b <basedn> -s <scope> <filter>
```

For more information on ACLs, refer to Chapter 14, "Access control" on page 395.

15.2.3 Authentication using SASL

The Simple Authentication and Security Layer (SASL) is a framework for multiple authentication and encryption mechanisms for connection-oriented protocols. It has been added to LDAP Version 3 to overcome the authentication shortcomings of LDAP Version 2. For more information on SASL, please refer to:

http://www.ietf.org/rfc/rfc2222.txt?number=2222

Overview of SASL

SASL is a method for adding authentication support to connection based protocols. In SASL, connection protocols such as LDAP, IMAP, and so on are represented by profiles; each profile is considered a protocol extension that allows the protocol and SASL to work together. Among these are IMAP4, SMTP, POP3, and LDAP. Each protocol that intends to use SASL needs to be extended with a command to identify an authentication mechanism and to carry out an authentication exchange. LDAP Version 3 includes such a command: ldap_sasl_bind() (and ldap_sasl_bind_s()). Optionally, a security layer can be negotiated to encrypt the data after authentication and thus ensure confidentiality. The IBM Tivoli Directory Server supports SASL authentication using the CRAM-MD5 (Challenge Response Authentication Mechanism with Message Digest 5), DIGEST-MD5 mechanisms, which transmits message digests rather than the passwords themselves over the network.

Note: The SASL mechanisms supported by the IBM Tivoli Directory Server can be obtained by the following search command:

```
ldapsearch -s base -b "" objectclass=* supportedsaslmechanisms
```

The key parameters that influence the security method used are:

► DN: This is the distinguished name of the entry a requester wants to bind as. This can be thought of as the user ID in a normal user ID and password authentication.

► Mechanism: This is the name of the security method that should be used. The IBM Tivoli Directory Server supports CRAM-MD5, DIGEST-MD5 and external mechanisms. There is also an anonymous mechanism available which enables authentication as the generic user anonymous. In LDAP, the most common mechanism used is SSL (or its successor TLS), which is provided as a so-called external mechanism.

► Credentials: This contains the arbitrary data that identifies the DN. The format and content of the parameter depend on the mechanism chosen. If it is, for example, the ANONYMOUS mechanism, it can be an arbitrary string or an e-mail address that identifies the user.

Through the SASL bind API function call (sometimes also referred to as certificate bind), LDAP client applications call the SASL protocol driver on the server, which, in turn, connects the authentication system named in the SASL mechanism to retrieve the required authentication information for the user. SASL can be seen as an intermediator between the authentication system and a protocol like LDAP.

There is no special configuration necessary on either side (client or server) to use SASL/CRAM-MD5 authentication. Applications simply request it by making the appropriate API call. Some minimum set up is required for the SASL/DIGEST-MD5 authentication mechanisms, which can be found in the *IBM Tivoli Directory Server Version 5.2 Administration Guide*, which can be found at:

http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html

> **Attention:** If CRAM-MD5/DIGEST-MD5 authentication mechanisms are being used, then the userpassword cannot be stored encrypted using one-way hash algorithms like CRYPT or SHA. It is because these authentication mechanisms require the userpassword in clear text and passwords encrypted using one way hash algorithms cannot be retrieved in clear text. But two way hash algorithms like IMASK can be used for encrypting and storing the passwords.

In case of EXTERNAL mechanism, the client sends an initial response with an authorization identity. The server uses information, external to SASL, to determine whether the client is authorized to authenticate as the authorization identity. If the client is so authorized, the server indicates successful completion of the authentication exchange; otherwise the server indicates failure.

The system providing this external information may be SSL or TLS (or IPSec, but not used in IBM Tivoli Directory Server). SSL and TLS are mentioned later in this chapter.

15.2.4 Kerberos

The IBM Tivoli Directory server supports Kerberos Version 1.3 servers, such as the IBM Network Authentication Service, for AIX servers and AIX 64-bit clients. Use the version of Kerberos included with your operating system for AIX 32-bit clients, Windows NT and Windows 2000 clients.

> **Note:** You must have a Kerberos client installed to use Kerberos authentication.

Under Network Authentication Service, a client (generally either a user or a service) sends a request for a ticket to the Key Distribution Center (KDC). The KDC creates a ticket-granting ticket (TGT) for the client, encrypts it using the client's password as the key, and sends the encrypted TGT back to the client. The client then attempts to decrypt the TGT, using its password. If the decryption is successful, the client retains the decrypted TGT, indicating proof of the client's identity.

The TGT, which expires at a specified time, permits the client to obtain additional tickets that give permission for specific services. The requesting and granting of these additional tickets does not require user intervention.

Network Authentication Service negotiates authenticated, optionally encrypted communications between two points on the network. It can enable applications to provide a layer of security that is not dependent on which side of a firewall either client is on. Because of this, Network Authentication Service can play a vital role in the security of your network.

You need to create an LDAP server servicename in the key distribution center (KDC) using the principal name ldap/<hostname>.<mylocation>.<mycompany>.com.

> **Note:** An environment variable "LDAP_KRB_SERVICE_NAME" is used to determine the case of the LDAP Kerberos service name. If the variable is set to 'LDAP' then the uppercase LDAP Kerberos service name is used. If the variable is not set, then the lowercase ldap is used. This environment variable is used by both the LDAP client and the server. By default this variable is not set.

Network Authentication Service provides the following components:

Key distribution center

The Key Distribution Center (KDC) is a trusted server that has access to the private keys of all the principals in a realm. The KDC is composed of two parts:

► Authentication Server (AS)
► Ticket Granting Server (TGS)

The AS handles initial client authentication by issuing a TGT. The TGS issues service tickets that can be used by the client to authenticate to a service.

Administration server

The administration server provides administrative access to the Network Authentication Service database. This database contains the principals, keys, policies, and other administrative information for the realm. The administration server allows adding, modifying, deleting, and viewing principals and policies.

Password change service

The password change service allows users to change their passwords. The password change service is provided by the administration server.

Client programs

Client programs are provided to manipulate credentials (tickets), manipulate keytab files, change passwords, and perform other basic Network Authentication Service operations.

Application programming interfaces (APIs)

Libraries and header files are provided to allow the development of secure distributed applications. The APIs provided are described in the Application Development Reference.

For further information on setting of Kerberos for use with the Directory Server and other information, you may refer the *IBM Tivoli Directory Server Version 5.2 Administration Guide*, which can be found at:

http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html

15.3 Password policy enforcement

Password policy, if enforced judiciously, enhances directory server security by forcing the directory users to have newer and complex passwords which are

difficult to guess and hence less prone to dictionary attacks, without causing much pain to the user.

15.3.1 Overview

Password Policy is a set of rules that controls how passwords are used and administered in IBM Tivoli Directory Server. The IBM Directory Server Password Policy is based on the IETF Password Policy Internet Draft. These rules are made to ensure that users change their passwords periodically, and that the passwords meet the organization's syntactic password requirements. These rules can also restrict the reuse of old passwords and ensure that users are locked out after a defined number of failed attempts.

All users except the directory administrator and the members of the administrative group are forced to comply with this password policy. The passwords for the administrator and members of the administrative group never expire and the accounts are never locked. The directory administrator and members of the administrative group have sufficient access control privileges to modify users' passwords and the password policy.

Starting with V5.1 of the IBM Directory Server, there is a new Directory Entry created for Password Policy with the DN *cn=pwdpolicy*. This entry resides at the root of all servers and this same Password Policy entry is replicated to all servers in the replication topology (If at all the topology exists). Password Policy has two separate types of attributes, the Password Policy Entry Attributes and Password Policy operational attributes that are associated with entries which contain a user password.

The Password Policy Entry Attributes with their default values are as follows:

▶ pwdAttribute: This is the attribute, identifying all the attributes in the directory server, on which the password policy rules would apply. Till the latest release that is, ITDS 52, userPassword is the only attribute to whom the password policy rules apply. In the future releases it is planned to make this field editable and also multivalued, so that the Customers may select the attributes to whom the password policy rules may apply.

▶ ibm-pwdPolicy: This attribute identifies whether password policy is turned on for this system. By default, password policy is turned off. Hence this attribute has a value of "FALSE". This attribute will hold a value "TRUE" when enabled.

▶ pwdMinAge: This attribute specifies the minimum period for which the userPassword for a specific user should be used. In other words, if you set the userPassword of entry "cn=user1,o=ibm,c=us" to user1 and you set pwdMinAge to 300 seconds, then cn=user1,o=ibm,c=us has to use this password for a minimum period of 300 seconds. cn=user1,o=ibm,c=us cannot change the password before 5 minutes from the time of

setting/resetting the password. The default value for this attribute is 0, which means there is no minimum age imposed.

► pwdMaxAge: This attribute specifies the maximum period for which the userPassword for a specific user can be used, where after the password would expire. In other words, if you set the userPassword of entry "cn=user1,o=ibm,c=us" to user1 and you set pwdMaxAge to 6 days, then cn=user1,o=ibm,c=us can use this password for a maximum period of 6 days. If the password happens to expire the administrator can reset the password to a new value and give it to the affected user. The default value for this attribute is 0 days, which means the password does not expire. The expected setting from the GUI is for the number of days, whereas it is stored internally in terms of the number of seconds and is also displayed in terms of seconds when queried from the command line.

That clearly indicates that the password for a user has be used in the period between pwdMinAge and pwdMaxAge.

► pwdInHistory: This attribute gives the total number of passwords to be stored in the password history. The default for this attribute is 0, indicating no passwords are to be stored in the pwdHistory.

► pwdCheckSyntax: This attribute holds three values:
  – The default value 0, indicating the syntax checking will not be enforced.
  – A value of 1, indicating the syntax checking will be done only in case the passwords are not encrypted.
  – A value of 2, indicating the syntax checking would be done, irrespective of whether the passwords are encrypted or not.

► pwdMinLength: This attribute specifies the minimum length the password should have. That is, if this attribute has a value of 5, and if you set the password of "cn=user1,o=ibm,c=us" to user then the access using this password should not be allowed. The default for this attribute is 0, indicating no minimum length is imposed.

► pwdExpireWarning: This attribute specifies the period before the password expiry that a warning be sent to a user, that his password is about to expire in the pwdExpireWarning time. that is, if you set this attribute to 1 day, then if the password is about to expire in the next 24 hours, a warning would be sent to the user in that regard. The default value for this attribute is 0 days, meaning no warnings will be sent. The GUI expects the value for this attribute in terms of the number of days, whereas the value is stored in seconds and displayed in seconds when queried from the command line.

► pwdGraceLoginLimit: This attribute specifies the maximum number of times a user is allowed to login after the password has expired. that is, if you set this value to 3 then after the password has expired, the user would be allowed to

login for maximum three times where after his account would be locked. The default value for this attribute is 0, indicating that authentication will fail if the password has expired.

► pwdLockout: This attribute specifies whether the password is to be locked in circumstances, where the password can be locked. For example, if a user attempts to login with a wrong password for more than the maximum allowed times then the password may be locked. However, if this attribute is set to "FALSE", which happens to be the default setting, then the password may still be used for a successful authentication. In other words, no intervention would be required by the Administrator to reset the password. The user can login to his account irrespective of the number of failed attempts.

► pwdLockoutDuration: This attribute specifies the total duration for which a password may be locked. This is also an attribute running on the parallel lines of pwdLockout. In the sense that, suppose you set this attribute to 300 seconds. If an account gets locked, the same will get unlocked after 300 seconds, without the intervention by the Administrator. The default for this attribute is 0, indicating the password cannot be used to authenticate until reset by an administrator.

► pwdMaxFailure: This attribute specifies the total number of failed login attempts to be granted to a user. That is, if you set this attribute to 3, then after three unsuccessful attempts to login, the password would be locked out. The default value for this attribute is 0, indicating that the accounts will not be locked on any number of failed attempts to login, and the value of pwdLockout will be ignored.

► pwdFailureCountInterval: This attribute specifies the period to clear the number of failed login attempts. that is, If you set this attribute to a value of 300 seconds, the current number of failed login attempts would be set to 0 only after 5 minutes. However if this attribute has a value 0, which being the default setting, the failure counter is only reset by a successful authentication.

► pwdMustChange: This attribute specifies if a user is supposed to change his password when he logs in. A value of "TRUE", which is the default one, specifies that the users must change their passwords after administrator reset. If this is set to FALSE then the users might continue using the same password as was given by the Administrator.

► pwdAllowUserChange: This attribute specifies if a user is allowed to change his password. A default value of "TRUE" is set to this attribute, which specifies that the users are allowed to change their passwords. A value of "FALSE" to this attribute specifies not to allow the users to change their passwords.

► pwdSafeModify: This attribute specifies if the user is required to provide his/her old password when requesting for a password change. A default value of "FALSE" specifies that the user does not need to send their existing

password when doing a modify. If set to "TRUE" the old password is required to change the password to a new value.

► passwordMinAlphaChars: This attribute specifies the total number of alphabets that are to appear in the password. A default value of 0, indicates that the minimum number of alphabetic characters required in the password is 0. that is, even passwords like "1234" should be acceptable.

► passwordMinOtherChars: This attribute specifies the total number of non-alphabets that are to appear in the password. A default value of 0, indicates that the minimum number of numeric and special characters required in the password is 0. That is even passwords like "abcd" should be acceptable.

► passwordMaxRepeatedChars: This attribute specifies the total number of repetitions of the characters allowed in a password. The default value of 0, indicates that repetition of characters is allowed to any extent. For example, aabb should be acceptable. Here there are 2 repetitions. If you were to set this attribute to >= 2 then also this password would have worked. However setting this attributes to 1 wouldn't have worked.

► passwordMinDiffChars: This attribute specifies the total number of characters in a password that should differ from all the passwords existing in the password history. The default value of 0, indicates no limitation. For example, if you have the passwords abcd and efgh in the history, and you set the passwordMinDiffChars to 2. Now if you attempt to set a new password as abcf then this will not be allowed as the number of characters differing between abcf and abcd is 1 character. A minimum difference of 2 characters is expected.

In order to set/unset the password policy attributes, there are two ways. You can do these tasks using the webadmin or you may go through the command line. We will see both the options. In order to view the password policy attributes throughout the Webadmin:

► Connect to the LDAP server using the WebAdmin.

► Click **Server Administration**.

► Click the **Manage security properties** category.

► There are three links in the Main Area, which help you set/unset different attributes pertaining to password policy.

► Select the **Password policy** tab. Refer to Figure 15-1 on page 442 for the screen that is shown.
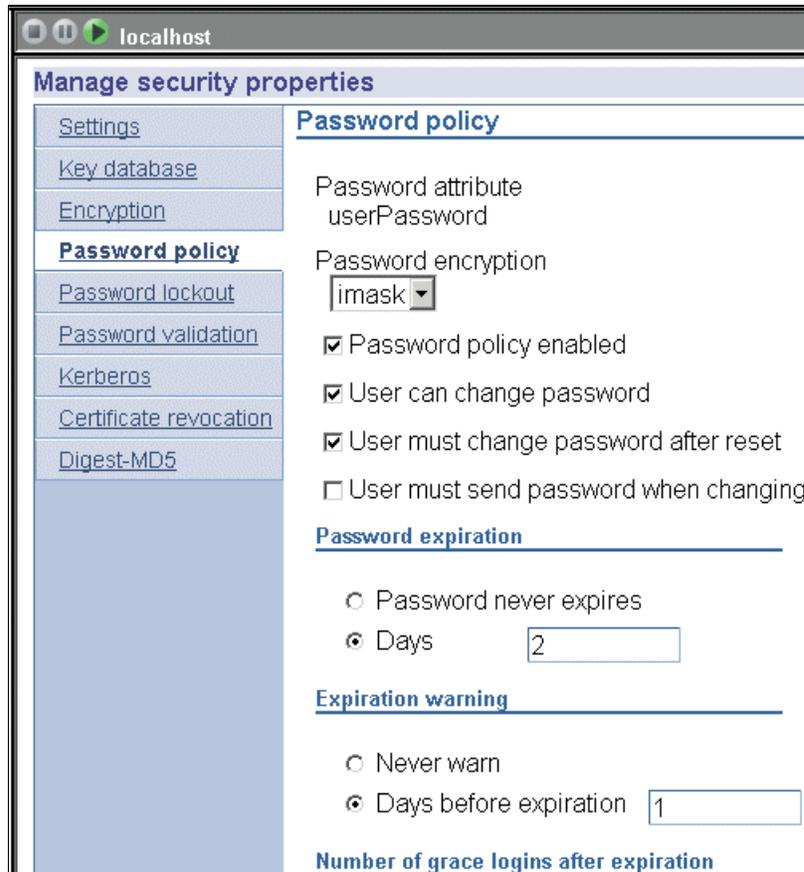
Figure 15-1   Set of attributes pertaining to Password policy

– As seen in Figure 15-1, the attribute affected by the password policy rules is userPassword.

– The current password encryption is set to imask. We will see more of this encryption settings in one of our subsequent sections.

– Check box corresponding to the Password policy enabled is checked, which indicates that password policy is enabled for this server.This attribute corresponds to the ibm-pwdPolicy attribute.

– Check box corresponding to the User can change password is checked, which indicates that the user is given access to change his password. This corresponds to the pwdAllowUserChange attribute.

– Check box corresponding to the User must change password after reset is checked, which indicates the user is forced to change the password once

reset by the administrator. This corresponds to the attribute pwdmustchange.

– Check box corresponding to the User must send password when changing is unchecked, which indicates the user need not send his passwords while changing his password. This corresponds to the pwdSafeModify attributes.

– Against Password expiration we have got a radio button to choose between Password never expires and Days. The setting here indicates a password expiration of 2 days. This setting corresponds to the pwdMaxAge attribute.

– Against Expiration warning we have got a radio button to choose between Never warn and Days before expiration. The setting here specifies that a warning be sent to the user 1 day prior to the expiration. This corresponds to the pwdexpirewarning attribute.

– The next field is Number of grace logins after expiration. The value against this field here signifies 2 grace logins after expiration. This corresponds to the attribute pwdgraceloginlimit.

► Select the **Password lockout** tab, and the screen in Figure 15-2 on page 444 is shown.
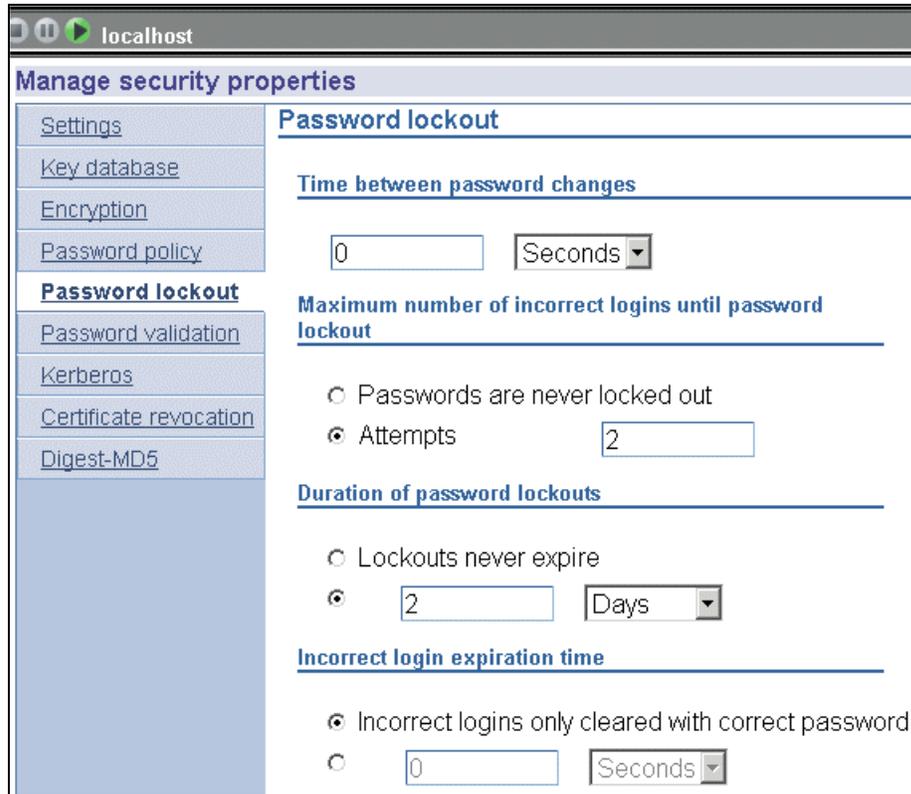
Figure 15-2   Set of attributes pertaining to Password lockout

– On the top the Time between password changes is to be specified. This is currently set to 0. This corresponds to the attribute pwdMinAge.

– The next field is a radio button for specifying the Maximum number of incorrect logins until password lockout. Here the setting is for a lockout after 2 failures. This corresponds to the pwdlockout attribute.

– The next radio button is where we specify Duration of password lockouts. Here the setting is for 2 days. of lockout after a password expiry. This corresponds to the attribute pwdlockoutduration.

– The next radio button is where we specify the Incorrect login expiration time. The settings here show that the incorrect logins are to be cleared only upon a successful authentication. This corresponds to the pwdfailurecountinterval.

▶ Select the **Password validation** tab, and the screen in Figure 15-3 on page 445 is shown.
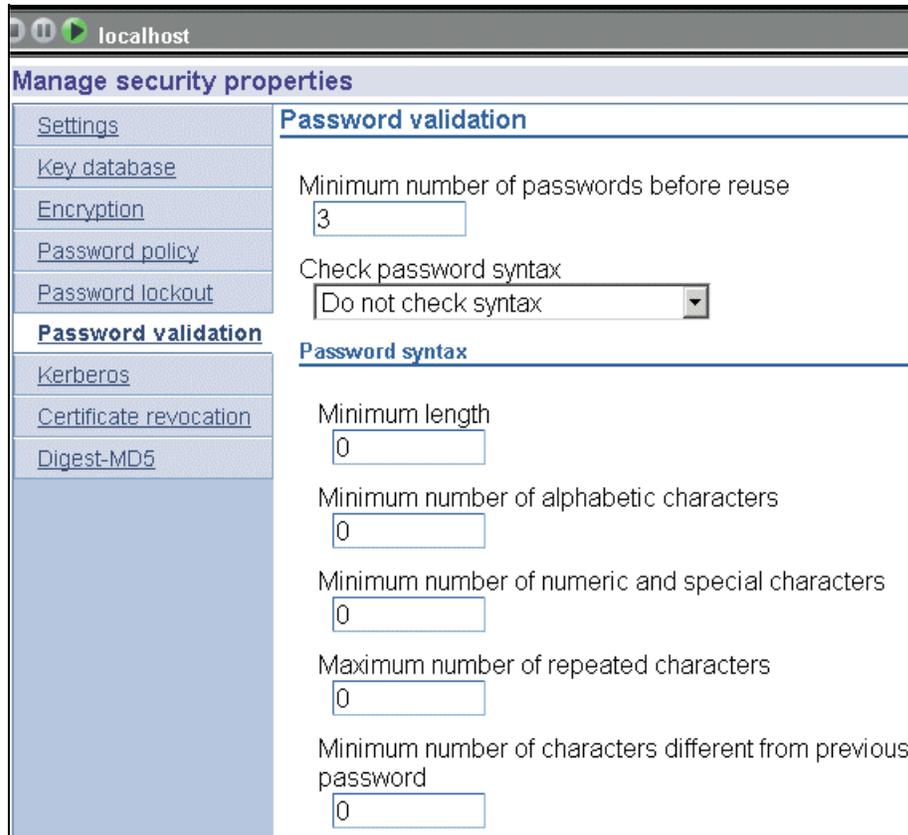
Figure 15-3   Set of attributes pertaining to Password validation

- On the top you specify the Minimum number of passwords before reuse. This is currently set to 3, indicating that we can't use the same password again till we have used 3 other passwords. This corresponds to the pwdinhistory attribute.

- Next we specify the value against Check password syntax. Here we have set the value to "Do not check syntax", which indicates that syntax checking is not to be done while evaluation password policy rules. This corresponds to the pwdCheckSyntax attribute.

- Next you specify the Minimum length of the password. This corresponds to the pwdMinLength attribute.

- Next you specify the Minimum number of alphabetic characters. This corresponds to the passwordMinAlphaChars attribute.

- Next you specify the Minimum number of numeric and special characters. This corresponds to the passwordMinOtherChars attribute.

- Next you specify the Minimum number of repeated characters.This corresponds to the passwordMaxRepeatedChars attribute.

- Next you specify the Minimum number of characters different from previous password. This corresponds to the passwordMinDiffChars attribute.

You can see the same attributes on the command line as follows:

```
D:\>ldapsearch -D cn=root -w secret -b cn=pwdpolicy objectclass=*
   cn=pwdpolicy
   objectclass=container
   objectclass=pwdPolicy
   objectclass=ibm-pwdPolicyExt
   objectclass=top
   cn=pwdPolicy
   pwdAttribute=userPassword
   pwdCheckSyntax=0
   pwdMinLength=0
   passwordMinAlphaChars=0
   passwordMinOtherChars=0
   passwordMaxRepeatedChars=0
   passwordMinDiffChars=0
   pwdSafeModify=false
   ibm-pwdpolicy=true
   pwdlockout=true
   pwdinhistory=3
   pwdgraceloginlimit=2
   pwdlockoutduration=172800
   pwdmaxfailure=2
   pwdallowuserchange=true
   pwdmustchange=true
   pwdexpirewarning=86400
   pwdmaxage=172800
   pwdminage=43
   pwdfailurecountinterval=23
```

If you want to change any of the attributes through command line, just create an LDIF and use the ldapmodify command. Here is an example. Suppose you want to change the password min age to 86 from the 43 shown above. Create an LDIF with the following contents:

```
dn: cn=pwdpolicy
changetype: modify
replace: pwdminage
pwdminage: 86
```

Suppose you name the above LDIF as pwdPolicy.ldif. Execute the ldapmodify command as:

```
ldapmodify -D <admin dn> -w <admin pw> -f pwdPolicy.ldif
```

That will do the necessary changes for you.

The above attributes were generic attributes pertaining to Password policy, applicable to the directory server as a whole. There are a set of operational attributes as well, which are set individually for each entry. These cannot be modified through direct client utilities. The server is supposed to modify them as and when needed.

The Password Policy Operational Attributes, which apply to any entry which contains a userPassword attribute are as follows:

► pwdChangedTime: This attribute specifies the last time the entry's password was changed. Here is an example of how this attribute is returned for an entry "cn=user1,o=ibm,c=us":

```
D:\>ldapsearch -D cn=root -w secret -b cn=user1,o=ibm,c=us objectclass=*
pwdChangedTime
    cn=user1,o=ibm,c=us
    pwdChangedTime=20040229231910.000000Z
```

This example shows that the password of cn=user1,o=ibm,c=us was modified on 29-02-2004 at 23:19:10 hours.

► pwdAccountLockedTime: This attribute holds the time that the user's account was locked. Here is an example of the same via the command line:

```
D:\>ldapsearch -D cn=root -w secret -b cn=user1,o=ibm,c=us objectclass=*
pwdAccountLockedTime
    cn=user1,o=ibm,c=us
    pwdAccountLockedTime=20040229232942.000000Z
```

This example shows that the account of cn=user1,o=ibm,c=us was locked on 29-02-04 at 23:29:04 hours.

Figure 15-4 on page 448 shows the screen you will see when you attempt to login when the relevant account is locked.
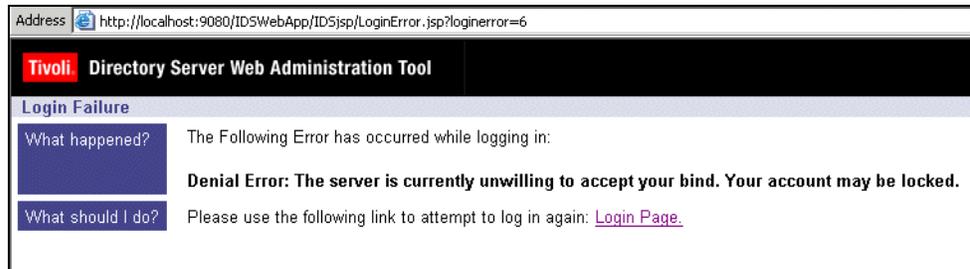
Figure 15-4   Account is locked

► pwdExpirationWarned: This attribute contains the time when the password expiration warning was first sent to the client. It will not show up any time in case the expiration warning was never sent. Here is an example of the expiration Warning message:

```
C:\>ldapchangepwd -D cn=user1,o=ibm,c=us -w user1 -n user
    ldap_simple_bind:  Warning, time before expiration is 58034
    changing password for entry cn=user1,o=ibm,c=us
```

The timestamp of the expiration warning sent, is stored in the same format as the timestamp for the other attributes, for example, as like pwdFailureTime.

► pwdFailureTime: This attribute holds the times of the consecutive authentication failures.

```
D:\>ldapsearch -D cn=root -w secret -b cn=user2,o=ibm,c=us objectclass=*
pwdFailureTime
    cn=user2,o=ibm,c=us
    pwdFailureTime=20040229235714.000000Z
```

This example shows that there has been only 1 login failure with regards user "cn=user2,o=ibm,c=us" and that was on 29-02-2004 at 23:57:14 hours.

► pwdHistory: This attribute holds a history of previously used passwords, the password portion of this attribute will be stored in the same encryption method as the userPassword is stored in. The passwords stored in this attribute will be compared to the new userPassword that the user has entered. Here is an example of looking up for pwdHistory:

```
D:\>ldapsearch -D cn=root -w secret -b cn=user2,o=ibm,c=us objectclass=*
userPassword pwdHistory
    cn=user2,o=ibm,c=us
    userPassword=user
    pwdHistory=20040301032149Z#2.5.4.35#171#{iMASK}>198o13ooQvIR95sxNtCDkCRi
    tZFPLyk8euKmCBz8OpJNEN8SZQVNtbGOqUMoQm3S9p3xVv+VQJGVOww2lx+lWPgDgAEIF1/S
    X98lvSFxiOjOXVNInK4ODOyTO5FGJ2unPP1+bM5CPanKf6VEdOlg7WONUzksFb4YwA<
    pwdHistory=20040301032308Z#2.5.4.35#33#{SHA}oYgcBu7JbbmQHHu/5BxCo/COnLQ=
    t
```

The above example shows that there are two encrypted passwords for the user cn=user2,o=ibm,c=us in the password history. Out of these one has an encryption of imask and the other has an encryption of sha. To learn more on the encryption level, please refer the section on the Server Encryption.

► pwdGraceUseTime: This attribute holds the timestamps of grace login once a password has expired, and is used to enforce the number of times an expired password may be used. If the grace logins are used then the timestamps will be stored in the same format as shown in the earlier password policy attributes above. Here is an example of the same:

```
C:\>ldapsearch -D cn=root -w secret -b cn=user2,o=ibm,c=us objectclass=*
pwdGraceUseTime
cn=user2,o=ibm,c=us
pwdGraceUseTime=20040303033651.000000Z
pwdGraceUseTime=20040303033711.000000Z
```

The above example shows that the user cn=user2,o=ibm,c=us had used two grace logins.

► pwdReset: This attribute holds a flag to indicates if the password has been reset.

Here is an example of the same:

```
D:\>ldapsearch -D cn=root -w secret -b cn=user1,o=ibm,c=us objectclass=*
pwdReset
    cn=user1,o=ibm,c=us
    pwdReset=true
```

The above example shows that the password for the user cn=user1,o=ibm,c=us was reset by administrator. Here is what is shown, when the password of the user is not reset:

```
D:\>ldapsearch -D cn=root -w secret -b cn=user3,o=ibm,c=us objectclass=*
pwdReset
    cn=user3,o=ibm,c=us
```

By means of enabling a setting in the directory server, it is possible to restrict the users from authenticating to the directory server when their password has been reset, unless they change their password. Figure 15-5 on page 450 shows the screenshot for password reset policy.

Figure 15-5   Policy pertaining to password reset

If the check box against "User must change password after reset" is checked and applied the users must change their password after the administrator has reset them or else clients are thrown back messages as shown in the following example:

```
D:\>ldapsearch -D cn=user3,o=ibm,c=us -w user -b cn=user3,o=ibm,c=us
objectclass=*
    ldap_simple_bind:  Error, Password must be changed after reset
    ldap_search:  DSA is unwilling to perform ---
    Error, Password must be changed after reset
```

Implementation

The Password Policy entry cn=pwdpolicy is created at the first server startup, if the entry is currently not present and the suffix for this entry resides in the IBM Directory Server config file.

In order to use Password Policy the Administrator must set the ibm-pwdpolicy in the cn=pwdpolicy entry to TRUE either by using the Web Administration Tools or doing an ldapmodify to modify the attribute. A set of details for configuring Password Policy using Web administration tool and command line has already been discussed above. However, if any further details are needed, please feel free to check out the *IBM Tivoli Directory Server version 5.2 Administration Guide* at:

http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html

All users may view the Password Policy entry but only the Administrator can modify it, by default. ACL's may be set to let other users modify the entry. The Password Policy operational attributes may be used in the filter when performing a search but these attributes will not be displayed unless distinctly specified in the search and the binding user has permission to search on them. Some examples of the same are already shown in the above explanations.

For most of the cn=pwdpolicy entry attributes, a value of zero indicates that this feature of the policy is not being used. All of the integer valued attributes must be set to 0 or a positive integer and all of the attributes must have logical values in respect to the other related attributes.

> **Note:** If you try to set/reset an attribute which conflicts with the settings of the other password policy attributes, then the relevant updates will not be completed and an error message would be flashed to the user saying `Error: Some of the changes could not be saved`. This is the error message seen in the status bar of the WebAdmin.

Password policy replication

Replica servers, which contain the entries being affected by Password Policy but are read-only to the client, do not have the following operational attributes replicated from the Master Server: pwdFailureTime, pwdAccountLockedTime, pwdGraceUseTime and pwdExpirationWarned.

These Replica servers contain local copies of these operational attributes, but when the entry's userPassword attribute is modified on the Master Server, these local attributes are cleared as they are on the Master.

The LDAPCHANGEPWD Command

This is a new client tool that users can use to modify their password easily. It always performs a safe modify of the user's password in case the pwdSafeModify attribute is set to TRUE. The user must supply their DN, current password and new password. Further details on this tool can be seen in Chapter 10, "Client tools" on page 237.

15.4 Password encryption

Storing passwords in clear text in the backend has potential risks. Hence, IBM Tivoli Directory Server supports a function where the passwords can be encrypted before being stored in the directory. This prevents the passwords from being compromised via direct SQL queries, database file look-ups and unauthorized access. The directory administrator too is not authorized to see the

passwords in clear text although he has the right to change the password of any user.

The directory server can be configured to encrypt the password using one-way hash algorithms or two way hash algorithms. One-way hash algorithms used by the directory server are:

► SHA-1 (Secure Hash Algorithm)
► crypt

Passwords encrypted using one way hashes can be used for password matching but cannot be decrypted, that is, the clear text version of the password cannot be retrieved by applications. During user login, the login password is encoded and compared with the stored version for matching verifications. Hence, these passwords cannot be used by applications which require a clear text version of the password for authentication purposes.

For applications that require retrieval of clear passwords, such as middle-tier authentication agents, the directory administrator needs to configure the server to perform either a two-way encoding or no encryption on user passwords. In this instance, the clear passwords stored in the directory are protected by the directory ACL mechanism. The two-way hash algorithm used by the directory server is imask.

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

After the server is configured for using a given encryption algorithm, all new passwords (for newly added users) or modified passwords (for existing users) are encrypted using the given algorithm and then stored in the directory. The name of the algorithm is tagged to the encoded password so that passwords encrypted using different algorithms can co-exist. If the encryption algorithm is changed, the existing passwords remain unaffected and continue to work.

Here are a set of examples demonstrating the impact of the encryption levels.

In case encryption was set to imask when cn=user2,o=ibm,c=us was created.

```
C:\>ldapsearch -D cn=root -w secret -b cn=user2,o=ibm,c=us objectclass=*
userPassword
cn=user2,o=ibm,c=us
userPassword=user
```

In case now the encryption is changed to sha:

```
C:\>ldapsearch -D cn=root -w secret -b cn=user2,o=ibm,c=us objectclass=*
userPassword
cn=user2,o=ibm,c=us
userPassword=user
```

In case now we create a user cn=user4,o=ibm,c=us:

```
C:\>ldapsearch -D cn=root -w secret -b cn=user4,o=ibm,c=us objectclass=*
userPassword
cn=user4,o=ibm,c=us
userPassword={SHA}Et6pb+wgWTVmq3VpLJlJWWgzrck=
```

In case we set the encryption to crypt:

```
C:\>ldapsearch -D cn=root -w secret -b cn=user4,o=ibm,c=us objectclass=*
userPassword
cn=user4,o=ibm,c=us
userPassword={SHA}Et6pb+wgWTVmq3VpLJlJWWgzrck=
```

Now suppose we create a new user cn=user5,o=ibm,c=us:

```
C:\>ldapsearch -D cn=root -w secret -b cn=user5,o=ibm,c=us objectclass=*
userPassword
cn=user5,o=ibm,c=us
userPassword={crypt}m6nQvPf1lTjqI
```

Now suppose we switch back to imask:

```
C:\>ldapsearch -D cn=root -w secret -b cn=user5,o=ibm,c=us objectclass=*
userPassword
cn=user5,o=ibm,c=us
userPassword={crypt}m6nQvPf1lTjqI

C:\>ldapsearch -D cn=root -w secret -b cn=user4,o=ibm,c=us objectclass=*
userPassword
cn=user4,o=ibm,c=us
userPassword={SHA}Et6pb+wgWTVmq3VpLJlJWWgzrck=
```

We believe the above examples clearly hint what the behavior of the encryption algorithm is. While adding a new entry, the password encryption prevalent at that instant of time will be applicable to the entry's userPassword and that will be maintained even upon switching the encryption algorithm.

Note: To know the name of the encryption algorithm that was used to encrypt the password, take a dump of the directory data using db2ldif utility. The ldif file so created contains the encrypted password along with the name of the encryption algorithm tagged to it.

Using the Web Administration Tool/command line, the directory server can be configured for the following encryption options:

► None: No encryption. Passwords are stored in the clear text format.

► crypt: Passwords are encoded by the UNIX crypt encoding algorithm before they are stored in the directory.

► SHA-1: Passwords are encoded by the SHA-1 encoding algorithm before they are stored in the directory.

► imask: Passwords are encoded by the imask algorithm before they are stored in the directory and are retrieved as part of an entry in the original clear format. This is the default.

The screenshot shown in Figure 15-1 on page 442 shows the tab for changing the password through the Web Administration tool.

In addition to userPassword, values of the *secretKey* attribute are always "imask" encoded in the directory. Unlike userPassword, this encoding is enforced for values of secretKey. No other option is provided. The secretKey attribute is an IBM defined schema. Applications may use this attribute to store sensitive data that always needs to be encoded in the directory and to retrieve the data in clear text format using the directory access control.

> **Note:** When imask is used as the server password encryption method, only the first 46 characters of a password entered are effective. Any characters after the 46th character will be ignored and considered as matched. Similarly, if the UNIX crypt method is used, only the first eight characters will be effective. Also since the value of SecretKey is encrypted in the database using the imask encryption, the SecretKey values which are longer than 46 characters will not be maintained.

The attribute associated with directory password encryption in the config file is *ibm-slapdPwEncryption*. Its value can be dynamically updated (after changing using Web Administration Tool or command line ldapmodify) using the *ldapexop* command line tool.

Here is how ldapmodify is used to change the encryption algorithm:

```
C:\>ldapmodify -D <admin DN> -w <admin PW>
    dn: cn=Configuration
    changetype: modify
    replace: ibm-slapdPWEncryption
    ibm-slapdPWEncryption: sha
```

That will change the encryption to sha. You can use a file instead of providing everything on the command line. Once you have done the above update you can

ask the server to take into effect the above change dynamically, using the ldapexop tool. For more information on ldapmodify/ldapexop please refer to Chapter 10, "Client tools" on page 237.

> **Note:** It is not feasible to create a new attribute which would accept values in a masked form as like userPassword. The masking of characters in the field of userPassword is not based on the encryption algorithm chose, but it is the way that attribute is internally designed.

15.5 SSL/TLS support

The IBM Tivoli Directory Server has the ability to protect LDAP access by encrypting data with either Secure Sockets Layer (SSL) security or Transport Layer Security (TLS) or both. When using SSL or TLS to secure LDAP communications with the IBM Directory, both *server authentication* and *client authentication* are supported. To use SSL or TLS you must have GSKit installed on your system.

15.5.1 Overview of TLS

The primary goal of the TLS(Transport Layer Security)Protocol is to provide privacy and data integrity between two communicating applications. The protocol is composed of two layers: the TLS Record Protocol and the TLS Handshake Protocol.

TLS record protocol
It is the lower layer of TLS. It provides connection security with the following connection properties:

► Connection is private: It is ensured by using symmetric cryptography for data encryption (for example, DES, RC4, etc.). The keys for this encryption are uniquely generated per connection based on secrets negotiated by some other protocol like TLS Handshake protocol. It can be used without encryption also.

► Connection is reliable: Message transportation includes a message integrity check using a keyed MAC (Message Authentication Code). Secure hash functions (SHA or MD5)are used for MAC computations.

TLS handshake protocol
Allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data. It ensures that the peer's identity can

be authenticated using asymmetric, or public key cryptography (for example, RSA, DSS).

► The negotiation of a shared secret is secure: The negotiated secret key is unavailable to any eavesdropper.

► The negotiation is reliable: no attacker can modify the negotiation communication without being detected by the communicating parties.

For more information on TLS protocol, please visit the following Web site:

http://www.ietf.org/rfc/rfc2246.txt

Note: TLS is started by using the -Y option in the client tools.

TLS and SSL are not interoperable. Starting TLS over a SSL port gives operations error.

15.5.2 Overview of SSL

SSL is an industry-standard security protocol that uses symmetric-key and public-key cryptographic technology. Symmetric-key cryptography uses the same key to encrypt and decrypt messages. Public-key cryptography uses a pair of keys: a public key and a private key. Each server's public key is published, and the private key is kept secret. To send a secure message to the server, a client encrypts the message using the server's public key. When the server receives the message, it decrypts the message using its private key. Only the server can decrypt this message because the private key required to decrypt this message is available only with the server.

SSL provides three basic security services(the below steps are equally valid for TLS also):

► Mutual authentication: Mutual authentication is the process whereby the client and the server convince each other of (and prove) their identities. The client and server identities are encoded in public-key certificates. A public-key certificate contains the following components whereby the issuer, also known as a Certificate Authority (CA), is a trusted organization, such as RSA Data Security Inc. or Verisign Inc.:

– Subject's distinguished name
– Issuer's distinguished name
– Subject's public key
– Issuer's signature
– Validity period
– Serial number

Rather than mutual authentication, which provides for maximum security, many implementations only use server authentication.

► Message privacy: Message privacy is achieved through a combination of public-key and symmetric key encryption. All traffic between an SSL client and an SSL server is encrypted using a key and an encryption algorithm negotiated during session setup.

► Message integrity: The message integrity service ensures that SSL session traffic does not change while en route to its final destination. SSL uses a combination of public/private keys and hash functions to ensure message integrity.

If SSL is used by LDAP for secure communication, the SSL session is established first before the normal LDAP protocol conversation can start. The following events take place for establishing an SSL session:

1. The client and the server exchange hello messages to negotiate the encryption algorithm and hashing function (for message integrity) to be used for the SSL session.

2. The client and server exchange X.509 certificates to validate their identities (if client authentication is not requested, only the server sends its certificate). Certificates are verified by checking the correctness of format and validity dates and by verifying that the certificate bears the signature of a trusted Certificate Authority (CA).

3. The client randomly generates a set of keys that are used for encryption. The keys are encrypted using the server's public key and securely communicated to the server.

4. Encrypted communication can now start using the generated key for encryption and decryption.

For server authentication to function, the IBM Tivoli Directory server must have a digital certificate (based on the X.509 standard). This digital certificate is used to authenticate the IBM Tivoli Directory server to the client application(s). During the initial SSL handshake, the LDAP server supplies the client with its X.509 certificate. If the client validates the server's certificate, a secure, encrypted communication channel is established between the LDAP server and the client application.

If client and server authentication is to be used, both the LDAP server and the client application must have a digital certificate. The server's digital certificate is used to authenticate the LDAP server to the client application (for example, an application built with IBM's LDAP application development toolkit). Similarly, the client's digital certificate is used to authenticate the client to the LDAP server (in terms of SSL's strong authentication mechanism). During the initial SSL handshake, the LDAP server and the client exchange certificates for mutual

validation. After the client validates the server's certificate and the server validates the client's certificate, a secure encrypted communication channel is established between the LDAP server and the client application.

15.5.3 SSL utilities

The graphical utility gsk7ikm (IBM Key Management GUI) is provided for IBM AIX, Windows NT, and a number of other IBM and non-IBM platforms to manage SSL X.509v3 certificate databases (also known as keyring files or keyring databases). Its use is required to configure and use Secure Sockets Layer (SSL). The gsk7ikm utility replaces other utilities, like ikeyman, used with earlier versions of IBM SSL support. With the IBM Tivoli Directory Server (and associated clients), both client and server keyring files are managed with gsk7ikm.

The gsk7ikm utility, together with the SSL libraries, form the IBM SSL toolkit known as GSKit (Global Security Kit). GSKit provides the SSL protocol functions as well as a set of Certificate Management Services (CMS) functions. These CMS functions provide access to the certificate database (the keyring file) as well as functions such as validating client certificates (including Certificate Revocation List processing). The current version is GSKit Version 7, which supports SSL Version 3.0, C/C++ for clients and servers and Java for clients.

Since strong encryption (as provided by SSL) is controlled by export and other regulations in the U.S. and other countries, different versions of GSKit exist for different countries. While the installable options differ among these versions, the user interface and configuration steps are generally the same as described in the following sections.

> **Note:** Encryption technology is subject to government regulations in the U.S. and other countries. Such regulations have changed recently and may change in the future. Due to this, the SSL packaging and implementation may be different as the product rolls out or may change thereafter.

GSKIT installation

GSKit is an independent installable option required only when SSL security is to be used. GSKit might already be installed on your system if another application required it to be installed. GSKit is shipped with the IBM Tivoli Directory Server in the appropriate version for your country. Please check for and follow any installation instructions that came with the product. If you are using the ISMP for installing IBM Tivoli Directory Server, select GSKit component by checking it when asked. For native installation, you have to use the operating system provided commands(installp in AIX, pkgadd in Solaris, rpm in linux etc.) for installing GSKit.

gsk7ikm utility

This utility with its graphical user interface is used to manage certificates. The specific tasks you can perform with ikmgui include:

► Create a key pair and request a certificate from a CA.
► Receive a certificate into a keyring file.
► Change a keyring password.
► Show information about a key.
► Delete a key.
► Make a key the default key in the keyring file.
► Export a key.
► Import a key into the keyring file.
► Designate a key as a trusted root.
► Remove trusted root key designation.

To run gsk7ikm, you need to have the Java Development Toolkit (JDK version 1.4.1 is recommended) installed and the JAVA_HOME environment variable pointing to its root directory. Figure Figure 15-6 on page 460 shows the screen shot of GSKit when launched using the command gsk7ikm.
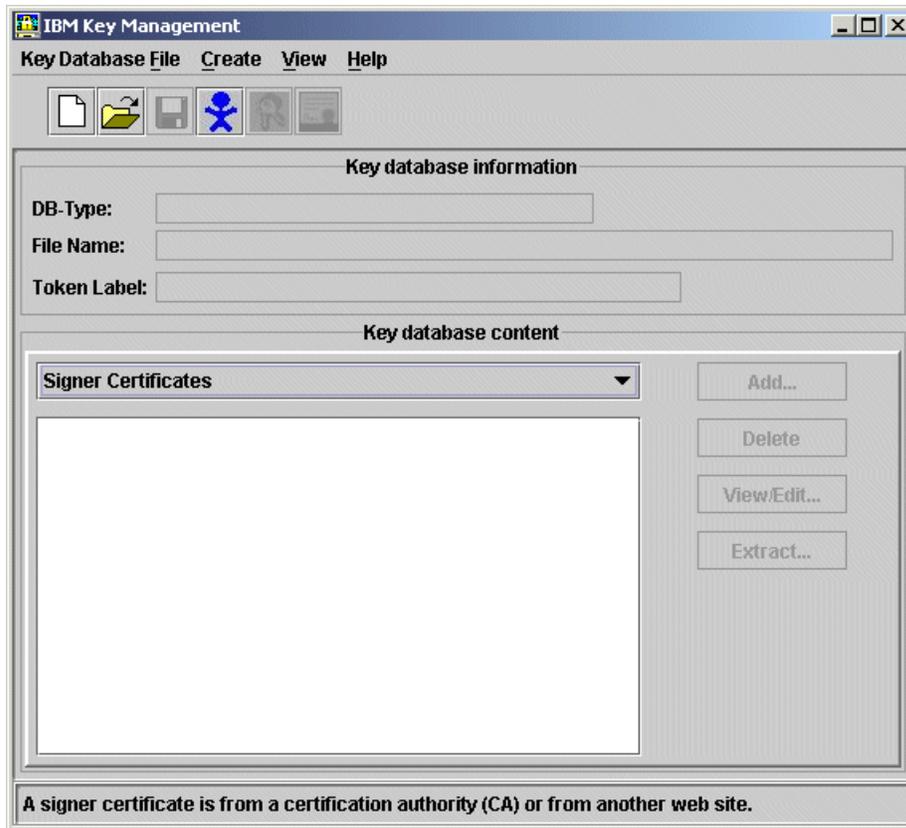
Figure 15-6   IBM Key Management tool

15.5.4 Configuring SSL security

To enable security with server authentication, you can follow one of the given steps:

► Create a certificate signed by a well-known certificate authority (CA): Create a public/private key pair and obtain and store a certificate from one of the predefined (well-known) Certificate Authorities. This procedure requires less setup because the keyring file is preconfigured with the CA root certificates required to identify the CAs from whom the certificate is issued.

► Create a self-signed certificate: The process of applying for and receiving a certificate from a CA can take two to three weeks. To enable SSL security until you receive the required CA root and server certificates, you can create a self-signed root certificate and store the certificate in the database and class files. To ensure maximum security for your site, you should only use a

self-signed certificate for server authentication until you receive a CA-issued certificate.

Creating a certificate signed by a trusted certificate authority

Using a certificate that was signed by a well-known (trusted) certificate authority gives you the advantage that most SSL communication partners know and trust that CA, and they will, therefore, most likely (depending on their configuration) accept a new certificate. This is especially helpful when communicating with partners outside your organization and beyond your authority to change security options. The disadvantages are that it takes some time (a few days or weeks) to get an official certificate and the fact that it is not for free. Creating a certificate signed by a well-known CA involves the creation of a key database and a certificate request that is then sent to the CA. After returning the certificate from the CA, it needs to be stored in the key database. These steps are detailed below using the GUI of the gsk7ikm utility.

1. Create a key database(.kdb file) for the server:

    a. Select **New** from the on Key Database File pull-down menu on the top of the main window (Figure 15-6 on page 460).

    b. On the dialog window that pops up, select **CMS key database** file in the Key database type selection list and then type in the name and location of the key database file to be created. This file has an extension of .kdb, as, for example, in ldap_key.kdb. Then, click **OK** to close the dialog panel.

    c. A new dialog pops up that requests your input for a password for the key database file, an optional expiration time, and whether or not the password is to be stashed to a file. Enter a password, an optional expiration time, and make sure that you check the check box next to **Stash the password to a file**? In case you are not stashing the password to a file, the password would be stored in the configuration file. Using a stashed password increases the level of secrecy. The server/applications would get to know the password by reading the stashed password file as and when needed. Consequently you need not mention the password in the communications with the server. Click **OK** to close this dialog. The password is then encrypted and stored in a file with the same name as database file but with an extension of .sth.

2. Create a certificate request.

    a. Select **New Certificate Request**… from the Create pull-down menu in the main window. In the dialog window that shows up, you will have to fill in the following information for the request:

        • Key label (a clear, descriptive label for the certificate)

        • Key size (512 or 1024, depending on security requirements and country version of the ikmgui utility)

- Common name

- Organization and other pertinent information to identify the owner of the certificate

- Full path of the file name for the certificate request file

b. Click **OK** to create the request. The .arm file so created contains the certificate request.

3. Send the certificate request, that is, the .arm file, to the certificate authority of your choice by mail or Web (follow their instructions, which can be found on their Web sites). (While you are waiting for the certificate authority to process and return your certificate, you can enable SSL security by creating, storing, and importing a self-signed certificate using the procedure described in the next section.) Once the certificate has been returned to you by the CA, you have to store it into the key database file.

4. Store the certificate into your database.

a. On the ikmgui main menu (Figure 15-6 on page 460), make sure that your key database file is open (check the filename in the Key database information portion of the window). If it is not open, choose **Open**... from the Key Database File pull-down menu and open your file.

b. Select **Personal Certificates** from the selection list in the lower Key database content portion of the window.

c. Click **Receive**... on the right of the window.

d. Supply the information about the file containing the signed certificate and click **OK**. This adds the certificate to the key database file. You will see the new certificate in the list under Personal Certificates.

5. A root certificate of the CA must be stored in the key database file. By default, root certificates of the most common CAs are already present in the file; so, you do not need to add them again. A trusted root is simply an X.509 certificate that has been signed by a trusted entity (for example, Verisign). You can see what root certificates there are by selecting **Signer Certificates** from the selection list in the Key database content portion of the main window. If your CA is not present in that list, obtain a root certificate from this CA and add it by clicking **Add**... on the right of the window.

Creating a self signed certificate

You can use the ikmgui utility to create a self-signed certificate to enable SSL sessions between clients and servers. The steps are essentially the same except that, in this case, you are the CA for the keys you will be creating, and you will be creating your own root certificate. The advantages of using this type of certificate is a quick start, it is free, and you have no dependencies on other organizations. The drawback, on the other hand, is that each client or server using this kind of

certificate needs to have the new root certificate imported, which may impose some administrative burden.

1. Create server key database (.kdb file).

   a. Click **Key Database File** (Figure 15-6 on page 460).

   b. Click New, from that dropdown that appears in point a above.

   c. On the dialog window that pops up, select **CMS key database** file in the Key database type selection list and then type in the name and location of the key database file to be created. This file has an extension of .kdb, as, for example, in ldap_key.kdb. Then, click **OK** to close the dialog panel.

   d. A new dialog pops up that requests your input for a password for the key database file, an optional expiration time, and whether or not the password is to be stashed to a file. Enter a password, an optional expiration time, and make sure that you check the check box next to Stash the password to a file? otherwise, you have to enter the password manually in the configuration file of the directory server. Click **OK** to close this dialog. The password is then encrypted and stored in a file with the same name as the key database file but with an extension of .sth.

2. Create a self-signed certificate.

   a. Select **New Self-Signed Certificate**... from the Create pull-down menu in the main window (Figure 15-6 on page 460). In the dialog window that shows up, you will have to fill in the following information:

      • Key label (a clear, descriptive label for the certificate)

      • Key Version (normally X.509 V3, unless you have reasons for other versions)

      • Key size (512 or 1024, depending upon security requirements and country version of the ikmgui utility)

      • Common name

      • Organization and other pertinent information to identify the owner of the certificate

      • Validity period in days

   Note: The key label and the organization are mandatory fields. The rest are optional.

3. Click **OK** to create the request. The .arm file so created contains the certificate request.

4. From the certificate just created above, you need to extract the root certificate that is necessary for other communication partners (clients and/or servers) to

recognize the newly created certificate. Here are the steps for exporting the root certificate:

a. Select the new certificate's entry in the Personal Certificate list and click **Extract Certificate** at the bottom right on the main window.

b. Select **Base64-encoded ASCII data** from the Data type list and enter a file name (with a .arm extension) and a location (directory) for the new root certificate to be exported to. Then click **OK** to export the root certificate. (If you want to create a file for the JNDI SSLight client key class, you must select **SSLight key database class** as data type when creating a file with a .class extension.)

You have now created a file that holds your own root certificate. This must be imported to all communication partners that will connect to the server through SSL.

5. Use the following steps for importing the new root certificate into others' key database (using ikmgui):

a. Make sure that the certificate extracted above, in the previous step, is made available to all the communication partners. You can transfer the file using ftp or a diskette or any other suitable media.

b. Invoke the ikmgui utility on the receiving system.

c. If not already done, create a key database file (see first step above for creating a self-signed certificate).

d. In the Key database content portion of the window, select **Signer Certificates** from the selection list and click **Add**... on the right.

e. Select **Base64-encoded ASCII data** from the Data type list and type the certificate file name and location into the appropriate fields. Then, click **OK** to import the certificate.

f. On the upcoming dialog, supply a label for this certificate and click **OK**.

The steps as described above need to be done on each machine that will communicate using this certificate with the machine from which the certificate was exported.

Each LDAP server should have its own certificate. Sharing certificates across multiple LDAP servers is not recommended. By using different certificates and private keys for each server, your security exposure is minimized should a keyring file for one of the servers be compromised.

Configuring the LDAP server to use SSL

After creating the key database files for the server, follow the steps given below for configuring the server to communicate over SSL:

1. Connect to the directory server using Web Administration Tool.

2. Click the **Server administration** tab and then select **Manage security properties.**

3. Click the **Settings** tab in the right pane. Select the type of secure connection and the type of authentication method you want.

4. Next click the **Key database** tab and provide the absolute path of your key database (.kdb) file.

5. If you have not stashed your password while creating the key database, you need to provide the password here.

6. From the Encryption tab, select the encryption algorithm. Multiple selections are allowed. If you select multiple encryption methods, the highest level of encryption is used by default; however, clients using the selected lower encryption levels still have access to the server.

7. If the Federal Information Processing Standards (FIPS) mode enablement feature is supported on your server, the Use FIPS certified implementation check box appears under the Implementation tab. If this check box is selected, the ICC library will be used for encryption. If you deselect the check box, a non-FIPS certified library will be used for encryption.

8. Restart the server for the changes to take effect. Also restart the Directory Administration daemon (ibmdiradm).

> **Note:** If the ibmdiradm daemon is not restarted, you will not be able to start or stop the ssl configured directory server from the Web administration tool.

Configuring the LDAP client to use SSL

There is no special setup required for LDAP clients using SSL other than the fact that the client must have the CAs root certificate in its key database file (see the steps described above). The application must then initiate a secure SSL connection by using the appropriate API calls, that is, ldap_ssl_client_init() in case of C applications. If client authentication is configured on the server, the client must be set up with its own certificate as described above for the server.

The command line tools supplied with IBM Tivoli Directory Server have special command line options to communicate with the server over SSL.

Here is an example of how you can fire a search to a server via SSL:

```
C:\>ldapsearch -D cn=root -w secret -Z -K F:\KEYS\clientCMS.kdb -P client -s
base objectclass=* | grep -i config
   namingcontexts=CN=CONFIGURATION
   ibm-configurationnamingcontext=CN=CONFIGURATION
   ibm-slapdisconfigurationmode=FALSE
```

The above query is a root DSE search over SSL.

```
The -Z flag is used to indicate that this is an SSL query.
The -K attribute is used to specify the path of the client key database.
The -P attribute is used to specify the password of the client key
database.
```

The server's certificate is stored in clientCMS.kdb and using this the client is able to tell the server that it is a valid client.

This is an illustration of the authentication method of "Server Authentication". On the same lines the "Server and Client Authentication" can be implemented. The way to implemented is almost the same way as described above. Above we see that the server exports its certificate to import it to the client. Similarly if the clients too export their certificates, which are imported into the server's key database file, then the corresponding authentication is known as the "Server and client authentication". In "Server and client authentication" both the server and the clients have a chance to verify that the request is coming from a valid client/server.

Configuring the Web administration tool to use SSL

The Web administration GUI tool is also a special type of client and hence needs some setup to communicate with an SSL configured server:

1. Start the Web Administration Tool and select the **Console Admin** from the drop down menu. Log in as the console administrator (the default username is superadmin and password is secret)

2. Click the **Console administration** tab and select **Manage console properties**.

3. Click **SSL key database**.

4. You need to have created a client key database of type jks and added the server certificate to it beforehand. Enter the absolute path of the jks key database file.

> **Note:** The procedure for creating a jks file and importing certificate is as like the procedure for the CMS databases explained above. The only difference being that while creating the database you give the type as either CMS for a CMS database or JKS for a JKS database.

5. If you have not stashed the password while creating the jks key database, enter the password in the password field.

6. Enter the absolute path for the Trust database file. Its usually the same as the keydatabase file.

7. Next click the **Manage console servers** tab. Select the appropriate server from the right panel and click **Edit**.

8. Change the Port to 636 and check the **SSL enabled** check-box.

9. Click **OK** to apply the changes.

10. The next time you log into this directory server from the Web administration console, all communication between the server and the console will be over ssl. If the server was configured for SSLonly mode and the corresponding changes were not made in the console, the console will fail to communicate with the sever.

Figure 15-7 show the screenshot for enabling the Web administration tool to access servers via SSL.
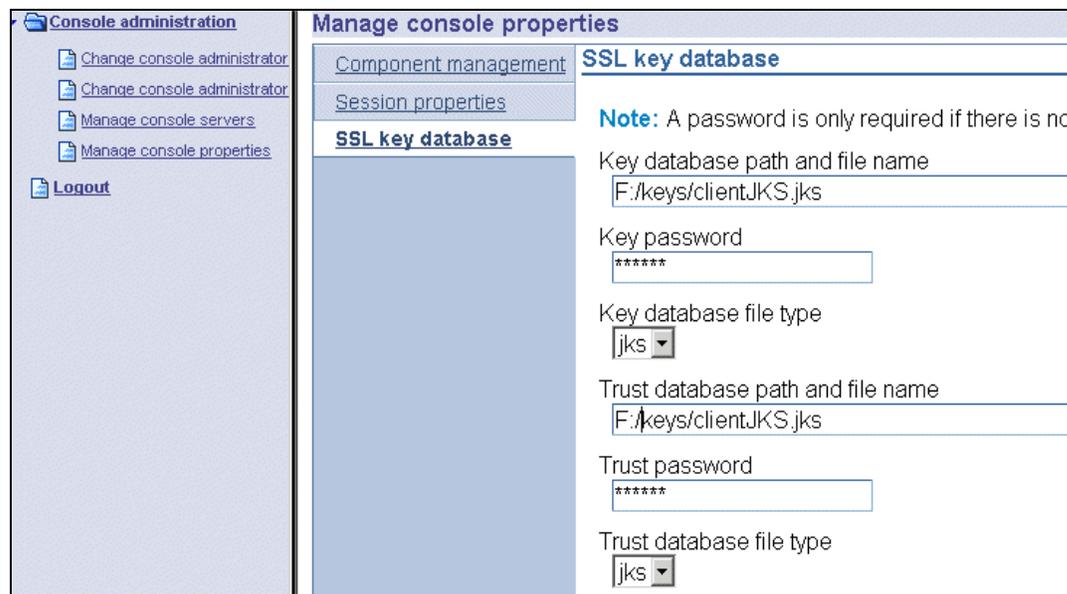
Figure 15-7   Enabling Webadmin to access servers via SSL

For more detailed steps for configuring the directory server and Web administration tool, please see the administration guide at:

`http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html`

> **Note:** If you wish to use the same server for both SSL and non-SSL communication then you can create an alias for the current server and configure the Webadmin to talk to the server over the non-SSL port. In this way you will have two entries for the same server, one for talking over SSL and the other for talking over non-SSL.

15.6 Protection against DoS attacks

Denial of Service (DoS) refers to a situation where the server is made to crash or is rendered unresponsive by bombarding it with huge number of client requests. These attacks cause huge losses to the victim in terms of time and money.

The IBM Tivoli Directory Server includes many advanced features to keep such attacks at bay. These features are listed below.

15.6.1 Non-blocking sockets

The read and write operations requested by the clients are handled intelligently so that a client which is trying to block the server by sending zero of partial data is automatically disconnected after a specified number of attempts.

On a read request, a client which continuously sends small amounts of data, is disconnected after a limited number of attempts. This limit is configurable and is represented by the attribute `ibm-slapdReadBlockedAttempts` in the config file under cn=Connection Management,cn=Frond End, cn=Configuration.

On a write request, a client is disconnected depending upon a write timeout value and the number of blocked write attempts. Both the values are configurable and are represented by the attributes `ibm-slapdWriteTimeout` and `ibm-slapdWriteBlockedAttempts` respectively in the config file under cn=Connection Management,cn=Frond End, cn=Configuration.

15.6.2 Extended operation for killing connections

This new extended operation, unbind, can be used by the administrator to terminate any faulty client connections and hence stop a probable DoS attack. This extended operation can be used to kill client connections that are:

► Bound to the server as a given DN

► Originated from a given client (IP Address)

► Bound to the server as a given DN and originated from a specified client

► Or all existing connections to the server

A *purge all connections* request would purge all connections except the connection the request came from. In all the above cases, connections not being served by a worker thread are terminated immediately. In case a worker is currently serving a connection, it is terminated once the operation is complete.

15.6.3 Emergency thread

A major serviceability problem occurs when all the worker threads of the server are busy performing some backend operations or are locked up. Even the directory administrator is unable to perform any diagnostic actions on the server. The emergency thread has been introduced to cope with such situations. It gets activated in case of such overloaded situations.

The emergency thread will be activated depending upon two configurable conditions:

► Size of work queue which represents the number of pending operations. This is represented by the variable `ibm-slapdESizeThreshold` in the config file.

► Time since the last item was removed from the queue, only if there are more items in the queue. This is represented by the attribute `ibm-slapdETimeThreshold` in the config file.

The admin can also specify which of the two, or a combination of the two will activate the emergency thread. This configurable parameter is represented by the attribute `ibm-slapdEThreadActivate` in the config file.

When the emergency thread is activated, a message will be logged into the ibmslapd.log file. The emergency thread will be deactivated when a worker removes an item from the work queue.

The emergency thread will support the following operations from the directory administrator:

► Kill Connection extended operation. It ensures that the admin will be able to stop a DoS attack without having to restart the server.

► Dynamic updates to the attributes in cn=Connection Management, cn=Front End, cn=Configuration section of the config file.

► Dynamic updates to the admin DN and admin password.

► Stopping and requesting the status of the server.

► Reading and clearing log files.

► Monitor and root DSE searches

► Modify and delete operations in the config backend.

15.6.4 Connection reaping

The connection reaping functionality has been enhanced to reap connections depending upon the type of authentication. There are three different thresholds specifying when:

► only anonymous connections will be reaped. It is represented by `ibm-slapdAnonReapingThreshold` attribute in the config file.

► only connections bound other than admin and replication connections will be reaped. It is represented by `ibm-slapdBoundReapingThreshold` attribute in the config file.

► all connections will be reaped. It is represented by `ibm-slapdAllReapingThreshold` attribute in the config file.

With all three thresholds only connections meeting the idle time out criteria will be reaped.

15.6.5 Allow anonymous bind

To add an extra level of protection the server will be able to reject anonymous bind requests. This will be a global setting for all backends. When dynamically updated to false it will trigger the unbind of all anonymous connections. This is represented by the attribute `ibm-slapdAllowAnon` in the config file.

Note: When anonymous binds are not allowed, TLS will not work.

Here is a screenshot of where exactly you set the above attributes:

1. Connect to the server using Web Administration tool.

2. Click **Server Administration**.

3. Click **Manage Connection Properties**.

4. On the right-hand side will be a panel which would provide you the options of changing attributes pertaining to the Connection termination reaping, etc., which were just discussed.

Figure 15-8 on page 471 shows the screenshot corresponding to the General tab.

Figure 15-8   Attributes pertaining to connections

Let us see the relation between the Web Admin attributes and the attributes in our configuration file (ibmslapd.conf):

► Allow anonymous connections corresponds to the attribute ibm-slapdAllowAnon.

► Cleanup threshold for anonymous connections corresponds to the attribute ibm-slapdAnonReapingThreshold.

► Cleanup threshold for authenticated connections corresponds to the attribute ibm-slapdBoundReapingThreshold.

► Cleanup threshold for all connections corresponds to the attribute ibm-slapdAllReapingThreshold.

► Idle timeout limit (in seconds) corresponds to the attribute ibm-slapdIdleTimeOut.

► Result timeout limit (in seconds) corresponds to the attribute ibm-slapdWriteTimeout.

Figure 15-9 on page 472 shows the settings of the emergency thread.

Figure 15-9   Attributes pertaining to the emergency thread

And here is the correlation of the attributes mentioned above and the configuration file:

► Enable emergency thread corresponds to the attribute ibm-slapdEThreadEnable.

► Pending request threshold corresponds to the attribute ibm-slapdESizeThreshold.

► Time threshold (in minutes) corresponds to the attribute ibm-slapdETimeThreshold.

► Criteria for emergency thread activation corresponds to the attribute ibm-slapdEThreadActivate.

15.7 Access control

Please refer to Chapter 14, "Access control" on page 395, for detailed description of Access Control Mechanism implemented in the IBM Tivoli Directory Server.

15.8 Summary

Let us summarize as to what we have seen in this chapter:

► We went through the different types of authentication:
  – Anonymous authentication
  – Basic authentication

- – SASL mechanisms
- – Kerberos

► We went through the password policy feature as to how it plays a key role in securing the user's passwords by enforcing a set of rules on them. We also went through the implementation of the password policy in ITDS 52.

► We went through the password encryptions that the directory server supports and understood the significance of each of them.

► We went through securing the server via SSL and TLS.

► We also studied the IBM's key management tool and how it is useful in generating and maintaining the keys and the relevant certificates.

► We went through the concept of Denial of Service (DoS) attack and studied the attributes which would help in preventing such attacks.

► We also got to know how ACLs play a vital role in the security of the directory entries/users.

16

Performance Tuning

This chapter describes some best practices for tuning your Lightweight Directory Access Protocol (LDAP). The IBM Tivoli Directory Server (ITDS) out of the box will fit most small directories, but for most enterprise directory examples you will need to do more tuning. This chapter goes over some of the main issues that come from Performance Tuning. We cover LDAP Cache, DB2 settings, and special OS-related settings that need to be addressed.

Performance Tuning is a art form; there is no cookie cutter approach that fits all directories for all occasions. But there are basic starting settings that can be made that will get you in the ball park.

The best friend to LDAP is memory; this is the one thing that can help right away with any directory.

► 32 bit OS memory limits are usually 4 GB depending on the OS.

► 64 bit OS memory limits are usually 16GB depending on the OS.

► See your own OS manufacturer to find out the limits that you have.

Tuning for optimal performance is primarily a matter of adjusting the relationships between the LDAP server and DB2 according to the nature of your workload. Because each workload is different, instead of providing exact values for tuning settings, guidelines are provided, where appropriate, for how to determine the best settings for your system.

Table 16-1   Tasks

Step	Task	See section
1	If the IBM Tivoli Directory server has never been started. Start it now to complete the server configuration. The initial DB2 database tables are not created until the first startup of the Directory server.	16.4.3
2	Optionally, back up the IBM Directory Server using DB2 backup. It is always a good idea to back up the IBM Directory Server before you make any major change. In this case, the change is performance tuning.	16.9.6
3	If this is a UNIX operating system, do the performance tuning tasks. These performance tuning tasks vary by operating system but they are mostly related to system resource limits.	16.10 for AIX 16.11 for Solaris
4	Check whether the IBM Directory Server change log is configured. By default it is not. It should only be used if required by other intergrator products. Performance is faster without the change log.	16.15.1
5	Check whether the IBM Directory Server audit-log is turned off. Performance is faster with the audit log turned off. Only use it for troubleshooting or if you require it for security monitoring.	16.15.2
6	Check to see if you are going to use Transaction and Event Notification; if you are not going to use these then turn them off. They are on by default.	16.3
7	Perform IBM Directory LDAP cache settings.	16.2
8	Perform slapd and ibmslapd conf file changes.	16.4
9	Perform the DB2 parameter performance tuning tasks.	16.5
10	Perform reorgchk and reorg of indexes and tables as needed.	16.7.2
11	Check to see if you have the needed indexes. Performance is greatly enhanced by having the right indexes in the DB2.	16.7.3
12	Use the monitoring outputs to help you make decisions on changes to the ldap and DB2 settings.	16.16

16.1 ITDS application components

Between a LDAP client and the server you have a network. One of the things you need to make sure of is how the network is put together. The biggest problem with networks is when you have different levels of speed and either Half duplex or Full duplex connections. Windows workstations most always default to auto setting; this can cause a number of problems in a network. You should always hard code your network cards to the speed and type, either half or full duplex connections. Do not ever use Auto mode. There will be times when windows boots up auto mode that it will not get the connection correctly and set you up in Half duplex mode when the switch or server is set to Full duplex and vice vesa. One tell-tail sign of this is that file transfer speeds are slow and there is high rate of collisions on an Ethernet network. If you are going to use Full duplex then every thing that talks to that server needs to be in Full duplex point to point. Have your network person check this out and make sure they hard code the workstations, switches, and servers to the same speed and connection type.

The query follows a path from the IBM Tivoli Directory Server client to the LDAP server, to DB2, to the physical disks in search of entries that match the query's search filter settings. The shorter the path to matching entries, the better overall performance you can expect from your system.

For example, if a query locates all the matching entries in the LDAP server within the LDAP Cache then access to DB2 and the disks is not necessary. If the matching entries are not found in the LDAP server cache, the query continues on to DB2 buffer pools and, if necessary, to the physical disks as a last resort. Because of the time and resources it takes to retrieve data from disk, it is better from a performance standpoint to allocate a significant amount of memory to the LDAP server caches and DB2 buffer pools.

16.2 ITDS LDAP caches

IBM Tivoli Directory Server LDAP caches and DB2 buffer pools store previously retrieved data and can significantly improve performance by reducing disk access. When requested data is found within a cache or buffer pool, it is called a cache click. A cache miss occurs when requested data is not located in a cache or buffer pool.

A cache miss is not necessary bad, it becomes bad when the miss rate continues to rise when the maximum cache level is reached. This says it has to go elsewhere to get it information.

Because the type of information in each cache and buffer pool is different, it is useful to understand why and when each cache is accessed. We will cover DB2 buffer pools in "DB2 tuning" on page 491.

16.2.1 LDAP caches

LDAP caches are fast storage buffers in memory used to store LDAP information such as queries, answers, and user authentication for future use. Tuning the LDAP caches is crucial to improving performance.

An LDAP search that accesses the LDAP cache can be faster than one that requires a connection to DB2, even if the information is cached in DB2. For this reason, tuning LDAP caches can improve performance by avoiding calls to the database.

The LDAP caches are especially useful for applications that frequently retrieve repeated cached information. Keep in mind that every workload is different, and some experimentation will likely be required in order to find the best settings for your workload.

Note that cache sizes for the filter cache, ACL cache, and entry cache are measured in numbers of entries.

LDAP Caches have changed over time with each major version change. In Version that are *older* then SecureWay 3.2.2 When there was any type of write to the database *all* the LDAP cache would be invalidated and would need to re-load its cache back again. This cause performance issues that would warrant you to not used any LDAP Cache. With SecureWay 3.2.2 and *newer* the only cache that would get invalidated would be Filter cache. This is still true today. But in future releases it is expected that the Filter cache would be smarter in how it invalidated itself. With the release of IDTS 5.2 a new Cache was put into the product called attribute cache. This will help this problem but it will not fix it. We will cover each of these LDAP caches in this section.

Changes for all these different caches are made in the LDAP config file. For version less then 3.2.2 it is called slapd.conf. For Versions 3.2.2 through 4.11 it is called slapd32.conf. And for 5.1 and 5.2 it is called ibmslapd.conf. For each of the different versions you can find this file in.../ldap/etc directory on all platforms.

Note: On Windows systems, the \etc\slapd32.conf or \etc\ibmslapd.conf file is not located at the root of the disk drive. You must search each disk to find it.

16.2.2 LDAP filter cache

When the client issues a query for data and the query first goes to filter cache. This cache contains cached entry IDs. There are two things that can happen when a query arrives at the filter cache:

► The IDs that match the filter settings used in the query are located in the filter cache. If this is the case, the list of the matching entry IDs is sent to the entry cache.

► The matching entry IDs are not cached in the filter cache. (cache miss) In this case, the query must access DB2 in search of the desired data. When it gets the information back from DB2 it will update the filter cache and the list of the matching entry ID's are sent to the entry cache. This will continue till you reach the maximum cache limits set in the ldap config file.

To determine how big your filter cache should be, run your workload with the filter cache set to different values and measure the differences in operations per second. This setting is measured in the number of entries and this is the maximum number of entries in the Filter Cache.

In 5.1 and later it is called:

`ibm-slapdFilterCacheSize: 25000`

In 4.1 and earlier it is called:

`ibm-slapdSetEnv: RDBM_FCACHE_SIZE=25000`

We will cover later how you can monitor this to see if it is reaching the maximum limits.

One thing to remember is that you want to minimize and batch updates (add, modify, modrdn, delete) when possible. This will lessen the problem with the filter cache being invalidated with any update.

There is no performance benefit in allocating any memory to the filter cache if even a small fraction of the operations in the workload are updates. If this proves to be the case for your workload, the only way to retain the performance advantage of a filter cache when updates are involved is to batch your updates. This allows long intervals during which there are only searches. If you cannot batch updates, specify a filter cache size of zero and allot more memory to other caches and buffer pools.

16.2.3 Filter cache bypass limits

The filter cache bypass limit configuration variable limits the number of entries that can be added to the filter cache. For example, if the bypass limit variable is

set to 1,000, search filters that match more than 1,000 entries are not added to the filter cache. This prevents large, uncommon searches from overwriting useful cache entries. To determine the best filter cache bypass limit for your workload, run your workload repeatedly and measure the throughput. Setting the limit too low downgrades performance by preventing valuable filters from being cached. Setting the filter bypass limit to approximately 100 appears to be the best size for most workloads. Setting it any larger benefits performance only slightly. If you set it to a value of "0" it is no limit. This setting is measured in the number of entries.

In 5.1 and later it is called:

`ibm-slapdFilterCacheBypassLimit: 100`

In 4.1 and earlier it is called:

`ibm-slapdSetEnv: RDBM_CACHE_BYPASS_LIMIT=100`

And with 4.1 and earlier you also have the following that will to the same with the entry cache:

`ibm-slapdSetEnv: RDBM_ENTRY_CACHE_BYPASS=YES`

If this variable is set (to anything) the entries associated with a search that matched more than RDBM_CACHE_BYPASS_LIMIT entries will also not be cached in the Entry cache.

With 5.1 and later they is no other entry to set this anymore.

16.2.4 LDAP entry cache

The entry cache contains cached entry data. Entry IDs are sent to the entry ache. If the entries that match the entry IDs are in the entry cache, then the results are returned to the client. If the entry cache does not contain the entries that correspond to the entry IDs, the query goes to DB2 in search of the matching entries. To determine how big your entry cache should be, run your workload with the entry cache set to different sizes and measure the differences in operations per second. You can use the cn=monitor command (this will be talked about later in this section. to help set this to a good level for your application.

This setting is measured in the number of entries and this shows the maximum number of entries in the Entry Cache.

In 5.1 and later it is called:

`ibm-slapdEntryCacheSize: 25000`

In 4.1and earlier it is called:

`ibm-slapdSetEnv: RDBM_CACHE_SIZE=25000`

16.2.5 Measuring filter and entry cache sizes

Filter cache and entry cache sizes are measured in numbers of entries. When determining how much memory to allocate to your LDAP caches, it can be useful to know how big the entries in your cache are. The following example shows how to measure the size of cached entries:

Note that this example calculates the average size of an entry in a sample entry cache, but the average filter cache entry size can be calculated similarly.

1. From the LDAP server:

   a. Set the filter cache size to zero.

   b. Set the entry cache size to a small value; for example, 200.

   c. Start ibmslapd (or slapd for 4.1 or earlier).

2. From the client:

   a. Run your application.

   b. Find the entry cache population (call this population1) using the following command:

      • On a Unix server:

        ```
        ldapsearch -h servername -s base -b cn=monitor objectclass=* | grep
        entry_cache_current
        ```

      • For a Windows server use the following command and search for entry_cache_current:

        ```
        ldapsearch -h servername -s base -b cn=monitor objectclass=*
        ```

3. From the LDAP Server:

   a. Find the memory used by ibmslapd (or 4.1 or earlier slapd - call this ibmslapd1):

      • On AIX operating systems, use ps v.

      • On Windows operating systems, use the VM size column in the Task Manager.

   b. Stop ibmslapd (on 4.1 and earlier slapd).

   c. Increase the size of the entry cache but keep it smaller than your working set.

   d. Start ibmslapd (on 4.1 and earlier slapd).

4. Run your application again and find the entry cache population (call this population2). See step 2b for the command syntax.

5. Find the memory used by ibmslapd (on 4.1 and earlier slapd - call this ibmslapd2). See step 3a for the command syntax.

6. Calculate the size of an entry cache entry using the following formula:

```
(ibmslapd size2 - ibmslapd size1) / (entry cache population2 - entry cache
population1)
```

For example, using this formula with the 500,000-entry database results in the following measurement:

```
(192084 KB - 51736 KB) / (48485 - 10003) = 3.65 KB per entry
```

16.2.6 LDAP ACL Cache

ACL Cache was not able to be changed till 4.1 and later. This is use to hold the users ACLs for that are in the LDAP. The current default setting should be enough for you needs. There is a monitor output that we will cover later in this section that you can see if it needs to be raised or not. There are two settings for ACL, one is if you want to use ACL cache or not and the other is to set the maximum ACL cache that can be used.

```
ibm-slapdACLCache: TRUE
ibm-slapdACLCacheSize: 25000
```

16.2.7 Setting other LDAP cache configuration variables

You can set LDAP configuration variables using the Web Administration Tool or the command line.

Using the Web Administration Tool

To set LDAP configuration variables using the Web Administration Tool:

1. Expand the Manage server properties category in the navigation area of the Web Administration tool.

2. Click **Performance**.

3. You can modify any of the following configuration variables:

   – Cache ACL information. This option must be selected for the Maximum number of elements in ACL cache settings to take effect.

   – Maximum number of elements in ACL cache (ACL cache size). The default is 25,000.

   – Maximum number of elements in entry cache (entry cache size). Specify the maximum number of elements in the entry cache. The default is 25,000.

   – Maximum number of elements in search filter cache (filter cache size).

The search filter cache consists of the requested search filters and resulting entry identifiers that matched. On an update operation, all filter cache entries are invalidated. The default is 25,000.

– Maximum number of elements from a single search added to search filter cache (filter cache bypass limit). If you select Elements, you must enter a number. The default is 100. Otherwise select Unlimited. Search filters that match more entries than the number specified here are not added to the search filter cache.

4. When you are finished, click **OK** to apply your changes, or click **Cancel** to exit the panel without making any changes.

Using the command line

To set LDAP configuration variables using the command line, issue the following command:

```
ldapmodify -D AdminDN -w Adminpassword -i filename
```

Where the file filename contains:

> **Note:** Make sure that there is a "-" between each attribute that is being changed on the same DN, and there is no space between each line of the same DN. There should only be one space between each DN.

```
dn: cn=Directory,cn=RDBM Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
changetype: modify
replace: ibm-slapdDbConnections
ibm-slapdDbConnections: 30
dn: cn=Front End, cn=Configuration
changetype: modify
replace: ibm-slapdACLCache
ibm-slapdACLCache: TRUE
-
replace: ibm-slapdACLCacheSize
ibm-slapdACLCacheSize: 25000
-
replace: ibm-slapdEntryCacheSize
ibm-slapdEntryCacheSize: 25000
-
replace: ibm-slapdFilterCacheSize
ibm-slapdFilterCacheSize: 25000
-
replace: ibm-slapdFilterCacheBypassLimit
ibm-slapdFilterCacheBypassLimit: 100
```

16.2.8 LDAP Attribute Cache (only on 5.2 and later)

To help with the problems that Filter Cache has with invalidating its cache with any update the attribute cache was built into ITDS 5.2. The attribute cache stores configured attributes and their values in memory. When a search is performed using a filter that contains all cached attributes, and the filter is of a type supported by the attribute cache manager, the filter can be resolved in memory. Resolving filters in memory leads to improved search performance.

When the client issues a query for some data, the first place the query goes is the attribute cache. There are two things that can happen when a query arrives at the attribute cache:

► All attributes used in the search filter are cached and the filter is of a type that can be resolved by the attribute cache manager. If this is the case, the list of matching entry IDs is resolved in memory using the attribute cache manager.

The attribute cache manager can resolve simple filters of the following types:

 – Exact match filters

 – Presence filters

The attribute cache manager can resolve complex filters only if they are conjunctive. In addition, the sub filters within the complex filters must be of the following types:

 – Exact match filters

 – Presence filters

 – Conjunctive filters

Filters containing attributes with language tags are not resolved by the attribute cache manager.

For example, if the attributes objectclass, uid, and cn are all cached, the following filters can be resolved in memory within the attribute cache manager:

 – (cn=Karla)

 – (cn=*)

 – (&(objectclass=eperson)(cn=Karla))

 – (&(objectclass=eperson)(cn=*)(uid=1234567))

 – (&(&(objectclass=eperson)(cn=*))(uid=1234567))

 – (&(uid=1234567)(&(objectclass=eperson)(cn=*)))

► Either some or all of the attributes used in the search filter are not cached or the filter is of a type that cannot be resolved by the attribute cache manager. If this is the case, the query is sent to the filter cache for further processing.

> **Note:** If there are no attributes in the attribute cache, the attribute cache manager determines this quickly, and the query is sent to the filter cache.
>
> For example, if the attributes objectclass, uid, and cn are the only cached attributes, the following filters will not be able to be resolved in memory by the attribute cache manager:
>
> ► (sn=Smith)
> ► (cn=K*)
> ► (|(objectclass=eperson)(cn~=Karla))
> ► (&(objectclass=eperson)(cn=K*)(uid=1234567))
> ► (&(&(objectclass=eperson)(cn<=Karla))(uid=1234567))
> ► (&(uid=1234567)(&(objectclass=eperson)(sn=*)))

► Determining which attributes to cache

To determine which attributes to cache, experiment with adding some or all of the attributes listed in the cached_attribute_candidate_click attribute to the attribute cache. Then run your workload and measure the differences in operations per second. Keep in mind that you only want to cache those attributes that are being used in your search's. This will take up more memory and it will be dynamically updated when there is a change in the values of those attributes that are cached.

16.2.9 Configuring attribute caching

You can configure attribute caching for the directory database, the changelog database, or both. Typically, there is no benefit from configuring attribute caching for the changelog database unless you perform very frequent searches of the changelog.

Using the Web Administration Tool

To configure the attribute cache using the Web Administration Tool:

1. Expand the Manage server properties category in the navigation area of the Web Administration Tool and select the Attribute cache tab.

2. You can change the amount of memory in bytes available to the directory attribute cache by changing the Directory cached attribute size (in kilobytes) field. The default is 16,384 KB (16 MB).

3. You can change the amount of memory in bytes available to the changelog attribute cache by changing the Changelog cached attribute size (in kilobytes) field. The default is 16,384 KB (16 MB).

> **Note:** This selection is disabled if a changelog has not been configured.

To add attributes to the attribute cache:

1. Select the attribute that you want to add as a cached attribute from the Available attributes menu. Only those attributes that can be cached are displayed in this menu; for example, sn.

   > **Note:** An attribute remains in the list of available attributes until it has been placed in both the cn=directory and the cn=changelog containers.

2. Click either **'Add to cn=directory'** or **'Add to cn=changelog'**. The attribute is displayed in the appropriate list box. You can list the same attribute in both containers.

   > **Note:** 'Add to cn=changelog' is disabled if a changelog has *not* been configured.

3. Repeat this process for each attribute you want to add to the attribute cache.

4. When you are finished, click **Apply** to save your changes without exiting, or click **OK** to apply your changes and exit, or click **Cancel** to exit this panel without making any changes.

Using the command line

To configure the attribute cache through the command line, issue the following command:

```
ldapmodify -D <adminDN> -w<adminPW> -i<filename>
```

Where <filename> contains the following, for example.

▶ For the directory database:

> **Note:** Make sure that there is a "-" between each attribute that is being changed on the same DN, and there is no space between each line of the same DN. There should only be one space between each DN.

```
dn: cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
changetype: modify
add: ibm-slapdCachedAttribute
ibm-slapdCachedAttribute: sn
-
```

```
add: ibm-slapdCachedAttribute
ibm-slapdCachedAttribute: cn
-
add: ibm-SlapdCachedAttributeSize
ibm-SlapdCachedAttributeSize: 16384
```

► For the changelog database:

> **Note:** Make sure that there is a "-" between each attribute that is being changed on the same DN, and there is no space between each line of the same DN. There should only be one space between each DN

```
dn: cn=changelog, cn=RDBM Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
changetype: modify
add: ibm-slapdCachedAttribute
ibm-slapdCachedAttribute: changetype
-
add: ibm-SlapdCachedAttributeSize
ibm-SlapdCachedAttributeSize: 16384
```

See the *IBM Tivoli Directory Server Version 5.2 Administration Guide* for more information.

http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html

16.3 Transaction and Event Notification

Transaction processing enables an application to group a set of entry updates together in one operation. Normally each individual LDAP operation is treated as a separate transaction with the database. Grouping operations together is useful when one operation is dependent on another operation because if one of the operations fails, the entire transaction fails. Transaction settings determine the limits on the transaction activity allowed on the server.

```
ibm-slapdTransactionEnable: TRUE
```

If you do not need to rely on one transaction to happen before another one or that each transaction stands on its own you will need to turn this off. This is done by setting this to FALSE:

```
ibm-slapdTransactionEnable: FALSE
```

Some other setting that deal with Transaction processing:

```
ibm-slapdMaxNumOfTransactions: 20
ibm-slapdMaxOpPerTransaction: 5
ibm-slapdMaxTimeLimitOfTransactions: 300
```

The *event notification* function allows a server to notify a registered client that an entry in the directory tree has been changed, added, or deleted. This notification is in the form of an unsolicited message. When an event occurs, the server ends a message to the client as an LDAP v3 unsolicited notification. The messageID is 0 and the message is in the form of an extended operation response. The responseName field is set to the registration OID. The response field contains the unique registration ID and a timestamp for when the event occurred. The time field is in UTC time format.

```
ibm-slapdEnableEventNotification: TRUE
```

If you do not need to have any event notification function then you will need to turn this off. This is done by setting this to FALSE:

```
ibm-slapdEnableEventNotification: FALSE
```

Some other setting that deal with event notification:

```
ibm-slapdMaxEventsPerConnection: 100
ibm-slapdMaxEventsTotal: 0
```

16.4 Additional slapd and ibmslapd settings

This section provides additional slapd and ibmslapd settings that can be used for tuning.

16.4.1 Tune the IBM Directory Server configuration file

In this section we discuss tuning the IBM Directory Server configuration file.

ibm-slapdSizeLimit

Change the slapd size limit for LDAP searches Edit either the slapd32.conf or the ibmslapd.conf configuration file, and then change the ibm-slapdSizeLimit parameter to a number other than 0 (unlimited). Setting this value affects all LDAP searches. If this value is set to unlimited (0), the time to compile and list all users increases, thus affecting system performance.

Tune this parameter so that a reasonable number is used for all LDAP searches. For example, setting ibm-slapdSizeLimit parameter affects the number of DN's that are listed by using the `ldapsearch` command.

ibm-slapdDbConnections and ibm-slapdSetEnv

Increase the number of IBM Directory Server connections to DB2 by editing the slapd32.conf or ibmslapd.conf file and increase the ibm-slapdDbConnections.

For AIX: With IDS 4.1 and later memory loopback is auto installed so you can use more then 256 meg of memory with LDAP. With using loopback you can increase the number of DB connections to 30. This used say in other documentation that this should be 15 or even 9. This was changed with memory loopback. With 3.2.2 and earlier without loopback you had a maximum of 8. If you manually configured for loopback then you could use 30+. But what we found in testing that not much was gained using anything grater then 32.

All other operating systems are set to 30.

If you do a migration from a older version to a newer version this sometimes keeps the old settings in the config file. So it is best to take a look at this and make sure it has been updated to the higher level.

The number of DB2 connections determines the amount of processing concurrency between the IBM Directory Server and DB2, one database connection will be established for each worker thread. If the number of DB2 connections is increased beyond its maximum value, the IBM Directory Server will revert to the maximum value.

With Secureway 3.2.2 and earlier you also will want to change the ibm-slapdSetEnv: SLAPD_WORKERS setting to match what your ibm-slapdDbConnections is. These need to match, what you don't want to happen is to have more worker threads then you have db connections. This will cause a performance issue with worker threads waiting for work to be done.

With IBM Directory 4.1 and later you only have ibm-slapdDbConnections this is also used for the number of worker threads.

16.4.2 Suffixes

Preferably, add only one suffix for all user directory objects. Example as follows:

```
dn: cn=Directory,cn=RDBM Backends,cn=IBMDirectory,cn=Schemas,cn=Configuration
ibm-slapdSuffix: localhost
ibm-slapdSuffix: user_suffix
```

Where user_suffix is the suffix to be used for user objects like o=ibm,c=us.

Note that it is recommended that only one suffix be used for user objects. You can separate the user namespace within the suffix by using multiple directory container objects. If more than one suffix is used, additional directory searches are necessary to find user objects, which slows down IBM Tivoli Directory server performance. For one or two additional suffixes, the performance slows down by approximately 10 percent.

Order the suffixes in the IBM Directory Server configuration file After the set of suffixes to be added has been determined, order them in the slapd32.conf or ibmslapd.conf file for best performance. The goal is to get the IBM Directory Server to return suffixes that are most likely to contain authenticating users first.

For ITDS Version 4.1 and earlier ldapsearch searches suffixes in the order in which the IBM Tivoli Directory server returns suffixes are returned in the reverse order as they appear in the configuration file.

For ITDS 5.x, suffixes are returned in the same order as they appear in the configuration file.

16.4.3 Recycle the IBM Directory Server

To recycle the IBM Directory Server to make it aware of any changes, do one of the following:

► To stop the IBM Directory Server, do the following:

On UNIX systems with ITDS Version 4.1 and earlier, enter the following:

```
ps –ef | grep slapd
# find the slapd process id
kill -9 slapd process id slapd
```

On UNIX systems with ITDS Version 5.1 and newer, enter the following:

```
ps –ef | grep ibmslapd
# find the ibmslapd process id
kill -9 ibmslapd process id ibmslapd
```

► To start the IBM Tivoli Directory server, do the following:

On UNIX systems with ITDS Version 4.1 and earlier, enter the following:

```
slapd
```

On UNIX systems with ITDS Version 5.1 and newer, enter the following:

```
ibmslapd
```

On Windows systems, stop and start the IBM Tivoli Directory Server service.

16.4.4 Verify suffix order

To verify that the suffixes are ordered for performance, enter the following on one line:

```
ldapsearch -h ldap_host -D cn=root -w ldap_passwd -s base -b "" "objectclass=*
```

Where cn=root is the IBM Directory Server root administrator user, ldap_host is the host name of the IBM Directory Server, and ldap_passwd is the IBM

Directory Server root administrator's password. you will get back the suffix order as how it is read along with some other server information.

There are several additional settings that affect performance by putting limits on client activity, minimizing the impact to server throughput and resource usage, such as:

With ITDS 5.1 and earlier:

► ibm-slapdTimeLimit: 900
► ibm-slapdIdleTimeOut: 300

With ITDS 5.2 and later:

► ibm-slapdPagedResAllowNonAdmin: TRUE
► ibm-slapdPagedResLmt: 3
► ibm-slapdSortKeyLimit: 3
► ibm-slapdSortSrchAllowNonAdmin: TRUE

Note: Default values are shown.

16.5 DB2 tuning

IBM Tivoli Directory Server uses DB2 as the data store and Structured Query Language (SQL) as the query retrieval mechanism. While the LDAP server caches LDAP queries, answers, and authentication information, DB2 caches tables, indexes, and statements. Best practices state that you should only have the LDAP instance on the DB2 that resides with the LDAP server. You should not share this DB2 with any other application. One reason is that the license that comes with the DB2 with IBM Tivoli Directory Server is only the license to be used by LDAP. The main reason not to share is that it will be a performance issue. You will be setting this DB2 to run LDAP and it is not a relational database so the settings will affect each other if you share the DB2 with other applications and not just the Directory Server.

Many DB2 configuration parameters affect either the memory (buffer pools) or disk resources. Since disk access is usually much slower than memory access, the key database performance tuning objective is to decrease the amount of disk activity.

We will be covering the following types of DB2 tuning:

► DB2 buffer pool tuning
► Other DB2 configuration parameters
► Optimization and organization (reorgchk and reorg)
► Backing up and restoring the database (backup and restore)

16.5.1 Warning when IBM Directory Server is running

DB2 parameter tuning commands make use of DB2 terminate. If the IBM Directory Server slapd or ibmslapd process is running when this command is issued, it will render the server partially functional. Any cached searches appear to respond correctly. Other searches might simply return with no results or error messages might appear. The recovery is to recycle the IBM Directory Server.

It is best to stop the IBM Directory Server when changing the DB2 tuning parameters.

For detailed information about IBM DB2 commands, see the IBM DB2 documentation at the following Web site:

http://www.ibm.com/software/data/db2/library/

> **Attention:** Only users listed as database administrators can run the DB2 commands. Be sure the user ID running the DB2 commands is a user in the dbsysadm group (UNIX operating systems) or a member of the Administrator group (Windows operating systems.) This includes the DB2 instance owner and root.

If you have any trouble running the DB2 commands, check to ensure that the DB2 environment variables have been established by running db2profile (if not, the db2 get and db2 update commands will not work). The script file db2profile is located in the sqllib subdirectory under the instance owner's home directory. If you need to tailor this file, follow the comments inside the file to set your instance name, user paths, and default database name (the default path is /home/ldapdb2/sqllib/db2profile.) It is assumed that the user is logged in as ibm-slapdDbUserId. If logged in as the root user on a UNIX operating system, it is possible to switch to the instance owner as follows:

```
su - instance_owner
```

Where *instance_owner* is the defined owner of the LDAP database.

To log on as the database administrator on a Windows 2000 operating system, run the following command:

```
runas /user:instance_owner db2cmd
```

The *instance_owner* is the defined owner of the LDAP database.

> **Note:** If you have problems connecting to the database on Windows systems, check the DB2INSTANCE environment variable. By default this variable is set to DB2. However, to connect to the database, the environment variable must be set to the database instance name.
>
> ```
> set DB2INSTANCE=instance_name
> ```
>
> The instance_name is by default called: ldapdb2.
>
> For most changes that you make you need to connect to the instance:
>
> ```
> db2 connect to database_name
> ```
>
> Where *database_name* is the name of the database. The default is ldapdb2.

For additional stability and performance enhancements, upgrade to the latest version of DB2.

> **Note:** Changes to DB2 configuration parameters do not take effect until the database is restarted with db2stop and db2start.

16.5.2 DB2 buffer pool tuning

DB2 buffer pool tuning is one of the most significant types of DB2 performance tuning. A buffer pool is a data cache between LDAP and the physical DB2 database files for both tables and indexes. DB2 buffer pools are searched when entries and their attributes are not found in the entry cache. Buffer pool tuning typically needs to be done when the database is initially loaded and when the database size changes significantly.

There are several considerations to keep in mind when tuning the DB2 buffer pools; for example:

► If there are no buffer pools, all database activity results in disk access.

► If the size of each buffer pool is too small, LDAP must wait for DB2 disk activity to satisfy DB2 SQL requests.

► If one or more buffer pools is too large, memory on the LDAP server might be wasted.

► If the total amount of space used by the LDAP caches and both buffer pools is larger than physical memory available on the server, operating system paging (disk activity) will occur.

To get the current DB2 buffer pool sizes, run the following commands:

```
db2 connect to database_name
```

```
db2 "select bpname,npages,pagesize from sysibm.sysbufferpools"
```

Where *database_name* is the name of the database.

Example 16-1 shows the default settings for the example above.

Example 16-1   Display DB2 buffer pool size default settings

```
BPNAME                NPAGES          PAGESIZE

IBMDEFAULTBP          29500           4096
LDAPBP               1230            32768

2 record(s) selected
```

This gives you appox 160 Mb of buffers used by the LDAP with in DB2.

With the start of SecureWay 3.2.2 and through IBM Tivoli Directory Server 5.2, the LDAP directory database (DB2) has two buffer pools: LDAPBP (32 K page size) and IBMDEFAULTBP (4 K page size). The size of each buffer pool needs to be set separately, but the method for determining how big each should be is the same: Run your workload with the buffer pool sizes set to different values and measure the differences in operations per second.

Note: DB2 does not allow buffer pools to be set to zero.

16.5.3 LDAPBP buffer pool size

This buffer pool contains cached entry data (ldap_entry) and all of the associated indexes. LDAPBP is similar to the entry cache, except that LDAPBP uses different algorithms in determining which entries to cache. It is possible that an entry that is not cached in the entry cache is located in LDAPBP.

To determine the best size for your LDAPBP buffer pool, run your workload with the LDAPBP buffer pool size set to different values and measure the differences in operations per second.

16.5.4 IBMDEFAULTBP buffer pool size

DB2 system information, including system tables and other information that is useful in resolving filters, is cached in the IBMDEFAULTBP buffer pool. You might need to adjust the IBMDEFAULTBP cache settings for better performance in the LDAPBP.

To determine the best size for your IBMDEFAULTBP buffer pool, run your workload with the buffer pool sizes set to different values and measure the differences in operations per second.

16.5.5 Setting buffer pool sizes

As a general guideline, a 3 to 1 ratio between memory allocated to the IBMDEFAULTBP (4-K pages) and LDAPBP (32-K pages) is good for performance. By default, the IBMDEFAULTBP is created with a size of 29500 (4-K) pages. By default, the LDAPBP buffer pool is created with a size of 1230 (32-K) pages.

On an LDAP Server with minimal memory configuration, this allocates roughly 60 percent of physical memory to the DB2 buffer pools

Use the alter bufferpool command to set the IBMDEFAULTBP and LDAPBP buffer pool sizes. The following examples show some IBMDEFAULTBP and LDAPBP buffer pools setting and what there real memory usage is:

► LDAP 3.2.2 and later defaults has memory usage: 154 MB

  – db2 alter bufferpool ibmdefaultbp size 29500
  – db2 alter bufferpool ldapbp size 1230

► 3 to 1 memory usage ratio has memory usage: 259.375 MB

  – db2 alter bufferpool ibmdefaultbp size 49800
  – db2 alter bufferpool ldapbp size 2075

► Doubling previous 3 to 1 ratio example has memory usage: 518.75 MB

  – db2 alter bufferpool ibmdefaultbp size 99600
  – db2 alter bufferpool ldapbp size 4150

► Doubling again has memory usage: 1037.5 MB

  – db2 alter bufferpool ibmdefaultbp size 199200
  – db2 alter bufferpool ldapbp size 8300

16.5.6 Warnings about buffer pool memory usage

If any of the buffer pools are set too high, DB2 can fail to start due to insufficient memory. If this occurs there might be a core dump file, but usually there is no error message.

On AIX systems, the system error log might report a memory allocation failure. To view this log, enter the following:

```
errpt –a | more
```

Restoring a database that was backed up on a system with buffer pool sizes that are too large for the target system might cause the restoration to fail.

If DB2 fails to start due to buffer pool sizes being too large, redo the DB2 tuning parameters.

16.5.7 Other DB2 configuration parameters

There are a number of other configuration settings that can be looked at exspecially some of the Heap settings. Functional problems can occur if one of the heap configuration parameters is to low or to hi. we have included some of the settings that are worth looking at. Some of these settings rely on other settings being set, these will be noted when that accrues.

Note: If DB2 recognizes that a parameter is configured insufficiently, the problem is posted to the diagnostic log (db2diag.log). We have included some of the error codes that you might get with some of these settings.

16.5.8 Warning about MINCOMMIT

Do not set the MINCOMMIT DB2 tuning to anything other than 1. In previous versions of documentation it has said to set the MINCOMMIT parameter to 25. A setting other than 1 might cause time-outs on update operations and might slow down the performance of these updates when you are using replication.

16.5.9 More DB2 configuration settings

In this section we discuss more DB2 configuration settings.

Utility Heap Size configuration parameter - util_heap_sz

Some information is:

► Default [Range]: 5000 [16 - 524 288]
► Unit of Measure: Pages (4 KB)
► When Allocated: As required by the database manager utilities
► When Freed: When the utility no longer needs the memory

This parameter indicates the maximum amount of memory that can be used simultaneously by the BACKUP, RESTORE, and LOAD (including load recovery) utilities.

Our recommendation is to use the default value unless your utilities run out of space, in which case you should increase this value. If memory on your system is constrained, you may wish to lower the value of this parameter to limit the

memory used by the database utilities. If the parameter is set too low, you may not be able to concurrently run utilities. You need to set this parameter large enough to accommodate all of the buffers that you want to allocate for the concurrent utilities. To set this heap to the following:

```
db2 update database configuration for ldapdb2 using UTIL_HEAP_SZ 5000
```

Application Control Heap Size configuration parameter - app_ctl_heap_sz

The information is:

▶ Database server with local and remote clients: 128 [1-64 000]
▶ Database server with local clients:
  – 64 [1-64 000] (for non-UNIX platforms)
  – 128 [1-64 000] (for UNIX-based platforms)
▶ Partitioned database server with local and remote clients: 512 [1-64 000]
▶ Unit of Measure: Pages (4 KB)
▶ When Allocated: When an application starts
▶ When Freed: When an application completes

This parameter specifies the average size of the shared memory area allocated for an application. There is one application control heap per connection per partition.

The application control heap is required primarily for sharing information between agents working on behalf of the same request.

This heap is also used to store descriptor information for declared temporary tables. The descriptor information for all declared temporary tables that have not been explicitly dropped is kept in this heap's memory and cannot be dropped until the declared temporary table is dropped.

Our recommendation is to initially start with the default value (128). You may have to set the value higher if you are running complex applications, if you have a system that contains a large number of database partitions, or if you use declared temporary tables. The amount of memory needed increases with the number of concurrently active declared temporary tables. A declared temporary table with many columns has a larger table descriptor size than a table with few columns, so having a large number of columns in an application's declared temporary tables also increases the demand on the application control heap. To set this heap to the following:

```
db2 update database configuration for ldapdb2 using APP_CTL_HEAP_SZ 128
```

If the APP_CTL_HEAP_SZ is set to an inadequate value, the following error message is issued when you import data into a database from shape files:

GSE0214N An INSERT statement failed. SQLERROR = SQL0973N Not enough storage is available in the "APP_CTL_HEAP" heap to process the statement.

Application Heap Size configuration parameter - applheapsz

The information is:

- ▶ Default [Range]

    - – 32-bit Database server with local and remote clients: 256 [16 - 60 000]
    - – 64-bit Database server with local and remote clients: 256 [16 - 60 000]
    - – 32-bit Partitioned database server with local and remote clients: 64 [16 - 60 000]
    - – 64-bit Partitioned database server with local and remote clients: 128 [16 - 60 000]

- ▶ Unit of Measure: Pages (4 KB)

- ▶ When Allocated: When an agent is initialized to do work for an application

- ▶ When Freed: When an agent completes the work to be done for an application

This parameter defines the number of private memory pages available to be used by the database manager on behalf of a specific agent or subagent.

The heap is allocated when an agent or subagent is initialized for an application. The amount allocated will be the minimum amount needed to process the request given to the agent or subagent. As the agent or subagent requires more heap space to process larger SQL statements, the database manager will allocate memory as needed, up to the maximum specified by this parameter.

Our recommendation is for most DB2 applications use an APPLHEAPSZ parameter value of at least 2048. Increase the value of this parameter if your applications receive an error indicating that there is not enough storage in the application heap.

The application heap (applheapsz) is allocated out of agent private memory.

If the APPLHEAPSZ is set to an inadequate value, the following error message is issued when you try to enable a database for spatial operations:

`GSE0009N Not enough space is available in DB2's application heap.`

`GSE0213N A bind operation failed. SQLERROR = SQL0001N Binding or precompilation did not complete successfully. SQLSTATE=00000.`

Sort Heap Size configuration parameter - sortheap

The information is:

- ▶ Default [Range]

    - – 32-bit platforms: 256 [16 - 524 288]
    - – 64-bit platforms: 256 [16 - 4 194 303]

- ▶ Unit of Measure: Pages (4 KB)
- ▶ When Allocated: As needed to perform sorts
- ▶ When Freed: When sorting is complete

This parameter defines the maximum number of private memory pages to be used for private sorts, or the maximum number of shared memory pages to be used for shared sorts. If the sort is a private sort, then this parameter affects agent private memory. If the sort is a shared sort, then this parameter affects the database shared memory. Each sort has a separate sort heap that is allocated as needed, by the database manager. This sort heap is the area where data is sorted. If directed by the optimizer, a smaller sort heap than the one specified by this parameter is allocated using information provided by the optimizer.

Our recommendation is 2500.

When working with the sort heap, you should consider the following: Appropriate indexes can minimize the use of the sort heap.

Hash join buffers and dynamic bitmaps (used for index ANDing and Star Joins) use sort heap memory. Increase the size of this parameter when these techniques are used.

Increase the size of this parameter when frequent large sorts are required.

When increasing the value of this parameter, you should examine whether the sheapthres parameter in the database manager configuration file also needs to be adjusted.

The sort heap size is used by the optimizer in determining access paths. You should consider rebinding applications (using the REBIND command) after changing this parameter.

SQL5155W means that the update completed successfully. The current value of SORTHEAP may adversely affect performance.

Explanation: The value of SORTHEAP is currently greater than half the value of the database manager configuration parameter SHEAPTHRES. This may cause performance to be less than optimal.

User Response: Increase the value of the database manager configuration parameter SHEAPTHRES and/or decrease the value of SORTHEAP so that SHEAPTHRES is at least twice as large as SORTHEAP.

A larger ratio is desirable in most cases. See the Administration Guide for recommendations on configuration parameter tuning.

SQL0955C: Sort memory cannot be allocated to process the statement. Reason code = reason-code.

Explanation:

► Insufficient virtual memory is available to the database agent for sort processing, as indicated by the reason code 1.

► Insufficient private process memory.

► Insufficient shared memory in the database-wide shared memory area designated for sort processing.

► The statement cannot be processed but other SQL statements may be processed.

User response: One or more of the following:

► Decrease the value of the sort heap parameter (sortheap) in the corresponding database configuration file.

► For reason code 1, increase the private virtual memory available, if possible. For example, on UNIX systems you can use the ulimit command to increase the maximum size of the data area for a process.

► For reason code 2, increase the size of the database-wide shared memory area designated for sort processing. To increase the size of this area without affecting the sortheap threshold for private sorts, increase the value of the SHEAPTHRES_SHR database configuration parameter.

► To increase both the size of the database-wide shared memory area designated for sort processing as well as the sortheap threshold for private sorting, increase the value of the SHEAPTHRES database manager configuration parameter and set SHEAPTHRES_SHR to 0.

SQL3537N: Sort memory could not be allocated during the execution of the LOAD utility.

Explanation: Insufficient process virtual memory is available to the LOAD utility for sort processing.

User Response: Terminate the application on receipt of this message. Ensure there is enough virtual memory available for sort processing.

Possible solutions include:

► Disconnect all applications from the database and decrease the size of the sort heap parameter (sortheap) in the corresponding database configuration file.

► Remove background processes and/or terminate other currently executing applications.

► Increase the amount of virtual memory available.

Sort Heap Threshold configuration parameter - sheapthres

The information is:

► Configuration Type: Database manager

► Default [Range]

  – UNIX 32-bit platforms: 20 000 [250 -- 2 097 152]
  – Windows platforms: 10 000 [250 -- 2 097 152]
  – 64-bit platforms: 20 000 [250 -- 2 147 483 647]
  – Unit of Measure: Pages (4 KB)

For shared sorts, this parameter is a database-wide hard limit on the total amount of memory consumed by shared sorts at any given time. When this limit is reached, no further shared-sort memory requests will be allowed.

The Sort Heap Threshold parameter, as a database manager configuration parameter, applies across the entire DB2 instance. The only way to set this parameter to different values on different nodes or partitions, is to create more than one DB2 instance. This will require managing different DB2 databases over different database partition groups. Such an arrangement defeats the purpose of many of the advantages of a partitioned database environment.

Recommendation: Ideally, you should set this parameter to a reasonable multiple of the largest sortheap parameter you have in your database manager instance. This parameter should be at least two times the largest sortheap defined for any database within the instance.

If you are doing private sorts and your system is not memory constrained, an ideal value for this parameter can be calculated using the following steps:

Calculate the typical sort heap usage for each database:

```
(typical number of concurrent agents running against the database)
multiplied by
(sortheap, as defined for that database)
```

Calculate the sum of the above results, which provides the total sort heap that could be used under typical circumstances for all databases within the instance.

You should use benchmarking techniques to tune this parameter to find the proper balance between sort performance and memory usage.

You can use the database system monitor to track the sort activity, using the post threshold sorts (post_threshold_sorts) monitor element. It is set by doing the following command:

```
db2 update dbm cfg using sheapthres 20000
```

Statement Heap Size configuration parameter - stmtheap

The information is:

- ► Default [Range]: 2048 [128 - 65 535]
- ► Unit of Measure: Pages (4 KB)
- ► When Allocated: For each statement during precompiling or binding
- ► When Freed: When precompiling or binding of each statement is complete

The statement heap is used as a work space for the SQL compiler during compilation of an SQL statement. This parameter specifies the size of this work space.

This area does not stay permanently allocated, but is allocated and released for every SQL statement handled. Note that for dynamic SQL statements, this work area will be used during execution of your program; whereas, for static SQL statements, it is used during the bind process but not during program execution.

Recommendation: In most cases the default value of this parameter will be acceptable. If you have very large SQL statements and the database manager issues an error (that the statement is too complex) when it attempts to optimize a statement, you should increase the value of this parameter in regular increments (such as 256 or 1024) until the error situation is resolved. This is the command you used to set it:

```
db2 update database configuration for ldapdb2 using stmtheap 2048
```

SQL0101N: The statement is too long or too complex.

Explanation: The statement could not be processed because it exceeds a system limit for either length or complexity, or because too many constraints or triggers are involved.

If the statement is one that creates or modifies a packed description, the new packed description may be too large for its corresponding column in the system catalogs.

Note that where character data conversions are performed for applications and databases running under different codepages, the result of the conversion is exceeding the length limit.

User response: Either:

- ► Break the statement up into shorter or less complex SQL statements.
- ► Increase the size of the statement heap (stmtheap) in the database configuration file.
- ► Reduce the number of check or referential constraints involved in the statement or reduce the number of indexes on foreign keys.
- ► Reduce the number of triggers involved in the statement.

SQL0437W: Performance of this complex query may be sub-optimal. Reason code: reason-code.

Explanation: The statement may achieve sub-optimal performance since the complexity of the query requires resources that are not available or optimization boundary conditions were encountered.

The following is a list of reason codes:

1. The join enumeration method was altered due to memory constraints.
2. The join enumeration method was altered due to query complexity.
3. Optimizer cost underflow.
4. Optimizer cost overflow.
5. Query optimization class was too low.
6. Optimizer ignored an invalid statistic.

The statement will be processed.

User Response: One or more of the following:

- ► Increase the size of the statement heap (stmtheap) in the database configuration file (Reason code 1).
- ► Break the statement up into less complex SQL statements (Reason codes 1,2,3,4).
- ► Ensure predicates do not over-specify the answer set (Reason code 3).
- ► Change the current query optimization class to a lower value (Reason codes 1,2,4).
- ► Issue Runstats for the tables involved in the query (Reason codes 3,4).
- ► Change the current query optimization class to a higher value (Reason code 5).

Reissue RUNSTATS for both the tables involved in the query and their corresponding indexes, that is, use the AND INDEXES ALL clause so that table and index statistics are consistent (Reason code 6)

Package Cache Size configuration parameter - pckcachesz

The information is:

▶ Default [Range]

– 32-bit platforms: -1 [-1, 32 -- 128 000]
– 64-bit platforms: -1 [-1, 32 -- 524 288]

▶ Unit of Measure: Pages (4 KB)

▶ When Allocated: When the database is initialized

▶ When Freed: When the database is shut down

This parameter is allocated out of the database shared memory, and is used for caching of sections for static and dynamic SQL statements on a database. In a partitioned database system, there is one package cache for each database partition.

Caching packages allows the database manager to reduce its internal overhead by eliminating the need to access the system catalogs when reloading a package; or, in the case of dynamic SQL, eliminating the need for compilation. Sections are kept in the package cache until one of the following occurs:

▶ The database is shut down.
▶ The package or dynamic SQL statement is invalidated.
▶ The cache runs out of space.

This caching of the section for a static or dynamic SQL statement can improve performance especially when the same statement is used multiple times by applications connected to a database. This is particularly important in a transaction processing application.

By taking the default (-1), the value used to calculate the page allocation is eight times the value specified for the maxappls configuration parameter. The exception to this occurs if eight times maxappls is less than 32. In this situation, the default value of -1 will set pckcachesz to 32.

Recommendation: When tuning this parameter, you should consider whether the extra memory being reserved for the package cache might be more effective if it was allocated for another purpose, such as the buffer pool or catalog cache. For this reason, you should use benchmarking techniques when tuning this parameter. If the cache is too large, memory is wasted holding copies of the initial sections.

The following monitor elements can help you determine whether you should adjust this configuration parameter:

- ► pkg_cache_lookups (package cache lookups)
- ► pkg_cache_inserts (package cache inserts)
- ► pkg_cache_size_top (package cache high water mark)
- ► pkg_cache_num_overflows (package cache overflows)

> **Note:** The package cache is a working cache, so you cannot set this parameter to zero. There must be sufficient memory allocated in this cache to hold all sections of the SQL statements currently being executed. If there is more space allocated than currently needed, then sections are cached. These sections can simply be executed the next time they are needed without having to load or compile them.

The limit specified by the pckcachesz parameter is a soft limit. This limit may be exceeded, if required, if memory is still available in the database shared set. You can use the pkg_cache_size_top monitor element to determine the largest that the package cache has grown, and the pkg_cache_num_overflows monitor element to determine how many times the limit specified by the pckcachesz parameter has been exceeded. The command for setting this is:

```
db2 update database configuration for ldapdb2 using pckcachesz 380
```

Statistics Heap Size configuration parameter - stat_heap_sz

The information is:

- ► Default [Range]: 4384 [1096 - 524 288]
- ► Unit of Measure: Pages (4 KB)
- ► When Allocated: When the RUNSTATS utility is started
- ► When Freed: When the RUNSTATS utility is completed

This parameter indicates the maximum size of the heap used in collecting statistics using the RUNSTATS command.

Recommendation: The default value is appropriate when no distribution statistics are collected or when distribution statistics are only being collected for relatively narrow tables. The minimum value is not recommended when distribution statistics are being gathered, as only tables containing 1 or 2 columns will fit in the heap.

You should adjust this parameter based on the number of columns for which statistics are being collected. Narrow tables, with relatively few columns, require less memory for distribution statistics to be gathered. Wide tables, with many columns, require significantly more memory. If you are gathering distribution statistics for tables which are very wide and require a large statistics heap, you

may wish to collect the statistics during a period of low system activity so you do not interfere with the memory requirements of other users.

Maximum Percent of Lock List Before Escalation config parameter - maxlocks

The information is:

- ▶ Default [Range]
- ▶ UNIX: 10 [1 - 100]
- ▶ Windows: 22 [1 - 100]
- ▶ Unit of Measure: Percentage

Lock escalation is the process of replacing row locks with table locks, reducing the number of locks in the list. This parameter defines a percentage of the lock list held by an application that must be filled before the database manager performs escalation. When the number of locks held by any one application reaches this percentage of the total lock list size, lock escalation will occur for the locks held by that application. Lock escalation also occurs if the lock list runs out of space.

The database manager determines which locks to escalate by looking through the lock list for the application and finding the table with the most row locks. If after replacing these with a single table lock, the maxlocks value is no longer exceeded, lock escalation will stop. If not, it will continue until the percentage of the lock list held is below the value of maxlocks. The maxlocks parameter multiplied by the maxappls parameter cannot be less than 100.

Recommendation: The following formula allows you to set maxlocks to allow an application to hold twice the average number of locks:

```
maxlocks = 2 * 100 / maxappls
```

Where 2 is used to achieve twice the average and 100 represents the largest percentage value allowed. If you have only a few applications that run concurrently, you could use the following formula as an alternative to the first formula:

```
    maxlocks = 2 * 100 / (average number of applications running
concurrently)
```

One of the considerations when setting maxlocks is to use it in conjunction with the size of the lock list (locklist). The actual limit of the number of locks held by an application before lock escalation occurs is:

```
maxlocks * locklist * 4096 / (100 * 36) on a 32-bit system
maxlocks * locklist * 4096 / (100 * 56) on a 64-bit system
```

Where 4096 is the number of bytes in a page, 100 is the largest percentage value allowed for maxlocks, and 36 is the number of bytes per lock on a 32-bit

system, and 56 is the number of bytes per lock on a 64-bit system. If you know that one of your applications requires 1 000 locks, and you do not want lock escalation to occur, then you should choose values for maxlocks and locklist in this formula so that the result is greater than 1 000. (Using 10 for maxlocks and 100 for locklist, this formula results in greater than the 1 000 locks needed.)

If maxlocks is set too low, lock escalation happens when there is still enough lock space for other concurrent applications. If maxlocks is set too high, a few applications can consume most of the lock space, and other applications will have to perform lock escalation. The need for lock escalation in this case results in poor concurrency.

You may use the database system monitor to help you track and tune this configuration parameter.

SQL5135N: The settings of the maxlocks and maxappls configuration parameters do not use all of the locklist space.

Explanation: The number of active processes (maxappls) times the maximum percentage of lock list space for each application (maxlocks) must be greater than or equal to 100. That is:

```
maxappls * maxlocks >= 100
```

This ensures that all of the allocated locklist space can be used.

User Response: Increase the setting for maxappls, maxlocks, or both.

Size of Log Files configuration parameter - logfilsiz
The information is:

▶ Default [Range]

  – UNIX: 1000 [4 -- 262 144]
  – Windows: 250 [4 -- 262 144]

▶ Unit of Measure: Pages (4 KB)

This parameter defines the size of each primary and secondary log file. The size of these log files limits the number of log records that can be written to them before they become full and a new log file is required.

The use of primary and secondary log files as well as the action taken when a log file becomes full are dependent on the type of logging that is being performed:

▶ Circular logging: A primary log file can be reused when the changes recorded in it have been committed. If the log file size is small and applications have processed a large number of changes to the database without committing the changes, a primary log file can quickly become full. If all primary log files

become full, the database manager will allocate secondary log files to hold the new log records.

► Log retention logging: When a primary log file is full, the log is archived and a new primary log file is allocated.

► Recommendation: You must balance the size of the log files with the number of primary and secondary log files.

The value of the logfilsiz should be increased if the database has a large number of update, delete and/or insert transactions running against it which will cause the log file to become full very quickly.

> **Note:** The upper limit of log file size, combined with the upper limit of the number of log files (logprimary + logsecond), gives an upper limit of 256 GB of active log space.

A log file that is too small can affect system performance because of the overhead of archiving old log files, allocating new log files, and waiting for a usable log file.

The value of the logfilsiz should be reduced if disk space is scarce, since primary logs are preallocated at this size.

A log file that is too large can reduce your flexibility when managing archived log files and copies of log files, since some media may not be able to hold an entire log file.

If you are using log retention, the current active log file is closed and truncated when the last application disconnects from a database. When the next connection to the database occurs, the next log file is used. Therefore, if you understand the logging requirements of your concurrent applications you may be able to determine a log file size which will not allocate excessive amounts of wasted space.

Recommendation: For most enterprise operations the default will not be enough this should be set to 10000. This can be set by the following command:

```
db2 update database configuration for ldapdb2 using LOGFILSIZ 10000
```

SQL1762N: Unable to connect to database because there is not enough space to allocate active log files.

Explanation: There is not enough disk space to allocate active log files. Possible reasons include the following.

There is insufficient space available on the device used to store the recovery logs.

If userexits are enabled, the userexit program may be failing due to an incorrect path, incorrect install directory, sharing violation, or other problem.

User Response: Based on the cause: Ensure that there is sufficient space on the device for the primary logs, as DB2 may require extra space to allocate new logs so that the database will start with at least LOGPRIMARY log files. Do *not* delete recovery logs to free space, even if they appear inactive.

Ensure the userexit program is operating correctly by manually invoking it. Review the instructions provided in the sample userexit source code for compiling and installing the userexit program. Ensure that the archive destination path exists.

As a last resort, try reducing the values for LOGPRIMARY and/or LOGFILSIZ database configuration parameters so that a smaller set of active log files are used. This will reduce the requirement for disk space.

Reissue the connect statement after determining and correcting the problem.

Number of Primary Log Files config parameter - logprimary

The information is:

▶ Default [Range]: 3 [2 - 256]
▶ Unit of Measure: Counter

When Allocated: The database is created a log is moved to a different location (which occurs when the logpath parameter is updated). Following a increase in the value of this parameter (logprimary), during the next database connection after all users have disconnected. A log file has been archived and a new log file is allocated (the logretain or userexit parameter must be enabled). If the logfilsiz parameter has been changed, the active log files are re-sized during the next database connection after all users have disconnected.

When Freed: Not freed unless this parameter decreases. If decreased, unneeded log files are deleted during the next connection to the database, otherwise you need to restart the database to free up space or set it up with a smaller primary log.

The primary log files establish a fixed amount of storage allocated to the recovery log files. This parameter allows you to specify the number of primary log files to be preallocated.

Under circular logging, the primary logs are used repeatedly in sequence. That is, when a log is full, the next primary log in the sequence is used if it is available. A log is considered available if all units of work with log records in it have been committed or rolled-back. If the next primary log in sequence is not available,

then a secondary log is allocated and used. Additional secondary logs are allocated and used until the next primary log in the sequence becomes available or the limit imposed by the logsecond parameter is reached. These secondary log files are dynamically deallocated as they are no longer needed by the database manager.

The number of primary and secondary log files must comply with the following:

```
If logsecond has a value of -1, logprimary <= 256.
If logsecond does not have a value of -1, (logprimary + logsecond) <= 256.
```

Recommendation: The value chosen for this parameter depends on a number of factors, including the type of logging being used, the size of the log files, and the type of processing environment (for example, length of transactions and frequency of commits).

Increasing this value will increase the disk requirements for the logs because the primary log files are preallocated during the very first connection to the database.

If you find that secondary log files are frequently being allocated, you may be able to improve system performance by increasing the log file size (logfilsiz) or by increasing the number of primary log files.

For databases that are not frequently accessed, in order to save disk storage, set the parameter to 2. For databases enabled for roll-forward recovery, set the parameter larger to avoid the overhead of allocating new logs almost immediately.

You may use the database system monitor to help you size the primary log files. Observation of the following monitor values over a period of time will aid in better tuning decisions, as average values may be more representative of your ongoing requirements.

```
sec_log_used_top (maximum secondary log space used)
tot_log_used_top (maximum total log space used)
sec_logs_allocated (secondary logs allocated currently)
```

Space requirements for log files. You will require 32 KB of space for log control files. You will also need at least enough space for your active log configuration, which you can calculate as:

```
(logprimary + logsecond) * (logfilsiz + 2) * 4096
```

Where logprimary is the number of primary log files, defined in the database configuration file. logsecond is the number of secondary log files, defined in the database configuration file; in this calculation, logsecond cannot be set to -1 (When logsecond is set to -1, you are requesting an infinite active log space.)

logfilsiz is the number of pages in each log file, defined in the database configuration file 2 is the number of header pages required for each log file 4096 is the number of bytes in one page. If the database is enabled for circular logging, the result of this formula will provide a sufficient amount of disk space.

If the database is enabled for roll-forward recovery, special log space requirements should be taken into consideration.

With the logretain configuration parameter enabled, the log files will be archived in the log path directory. The online disk space will eventually fill up, unless you move the log files to a different location.

With the userexit configuration parameter enabled, a user exit program moves the archived log files to a different location. Extra log space is still required to allow for:

▶ Online archived logs that are waiting to be moved by the user exit program
▶ New log files being formatted for future use

If the database is enabled for infinite logging (that is, you set logsecond to -1), the userexit configuration parameter must be enabled, so you will have the same disk space considerations. DB2(R) will keep at least the number of active log files specified by logprimary in the log path, so you should not use the value of -1 for logsecond in the above formula. Ensure you provide extra disk space to allow the delay caused by archiving log files.

If you are mirroring the log path, you will need to double the estimated log file space requirements.

See logfilsiz above for any errors that might come up other then the one below.

SQL5101N: The entries in the database configuration file define log file parameters (logprimary and logsecond) that are not in the valid range.

Explanation: The requested change would cause the total number of logfiles to be out of range. The following condition must always be true:

```
logprimary + logsecond <= 128
```

The requested change is not made.

User response: Do one or both of the following:

▶ Decrease the number of primary log files.
▶ Decrease the number of secondary log files.

Number of Secondary Log Files config parameter - logsecond

The information is:

- ▶ Default [Range]: 2 [-1; 0 - 254]
- ▶ Unit of Measure: Counter
- ▶ When Allocated: As needed when logprimary is insufficient (see detail below)
- ▶ When Freed: Over time as the database manager determines they will no longer be required.

This parameter specifies the number of secondary log files that are created and used for recovery log files (only as needed). When the primary log files become full, the secondary log files (of size logfilsiz) are allocated one at a time as needed, up to a maximum number as controlled by this parameter. An error code will be returned to the application, and the database will be shut down, if more secondary log files are required than are allowed by this parameter.

If you set logsecond to -1, the database is configured with infinite active log space. There is no limit on the size or the number of in-flight transactions running on the database. If you set logsecond to -1, you still use the logprimary and logfilsiz configuration parameters to specify how many log files DB2 should keep in the active log path. If DB2 needs to read log data from a log file, but the file is not in the active log path, DB2 will invoke the userexit program to retrieve the log file from the archive to the active log path. (DB2 will retrieve the files to the overflow log path, if you have configured one.) Once the log file is retrieved, DB2 will cache this file in the active log path so that other reads of log data from the same file will be fast. DB2 will manage the retrieval, caching, and removal of these log files as required.

If your log path is a raw device, you must configure the overflowlogpath configuration parameter in order to set logsecond to -1.

By setting logsecond to -1, you will have no limit on the size of the unit of work or the number of concurrent units of work. However, rollback (both at the savepoint level and at the unit of work level) could be very slow due to the need to retrieve log files from the archive. Crash recovery could also be very slow for the same reason. DB2 will write a message to the administration notification log to warn you that the current set of active units of work has exceeded the primary log files. This is an indication that rollback or crash recovery could be extremely slow.

To set logsecond to -1 the userexit configuration parameter must be set to yes.

Recommendation: Use secondary log files for databases that have periodic needs for large amounts of log space. For example, an application that is run once a month may require log space beyond that provided by the primary log files. Since secondary log files do not require permanent file space they are

advantageous in this type of situation. You will be best to set this to a high level or -1 to allow for growth when needed, such when you are loading a lot of users or making big replication loads. One thing you do not what to do is run out of log space or your database can be damage with lost or corrupted data. You can change this settings with this command:

```
db2 update database configuration for ldapdb2 using LOGSECOND 50
```

Change the Database Log Path config parameter - newlogpath

Default [Range]: Null [any valid path or device]

This parameter allows you to specify a string of up to 242 bytes to change the location where the log files are stored. The string can point to either a path name or to a raw device. If the string points to a path name, it must be a fully qualified path name, not a relative path name.

> **Note:** In a partitioned database environment, the node number is automatically appended to the path. This is done to maintain the uniqueness of the path in multiple logical node configurations

If you want to use replication, and your log path is a raw device, the overflowlogpath configuration parameter must be configured.

To specify a device, specify a string that the operating system identifies as a device. For example:

► On Windows: \\.\d: or \\.\PhysicalDisk5

> **Note:** You must have Windows NT Version 4.0 with Service Pack 3 or later installed to be able to write logs to a device.

► On UNIX-based platforms: /dev/rdblog8

> **Note:** You can only specify a device on AIX, Windows 2000, Windows NT, Solaris Operating Environment, HP-UX, and Linux platforms.

The new setting does not become the value of logpath until both of the following occur:

► The database is in a consistent state, as indicated by the database_consistent parameter.

► All users are disconnected from the database.

When the first new connection is made to the database, the database manager will move the logs to the new location specified by logpath. There might be log files in the old log path. These log files might not have been archived. You might need to archive these log files manually. Also, if you are running replication on this database, replication might still need the log files from before the log path change. If the database is configured with the User Exit Enable (userexit) database configuration parameter set to Yes, and if all the log files have been archived either by DB2 automatically or by yourself manually, then DB2 will be able to retrieve the log files to complete the replication process. Otherwise, you can copy the files from the old log path to the new log path.

If logpath or newlogpath specifies a raw device as the location where the log files are stored, mirror logging, as indicated by mirrorlogpath, is not allowed. If logpath or newlogpath specifies a file path as the location where the log files are stored, mirror logging is allowed and mirrorlogpath must also specify a file path.

Recommendation: Ideally, the log files will be on a physical disk which does not have high I/O. For instance, avoid putting the logs on the same disk as the operating system or high volume databases. This will allow for efficient logging activity with a minimum of overhead such as waiting for I/O. By Default the LDAP Directory is put on the same disk as the LDAP instance so you should move this to another disk.

You may use the database system monitor to track the number of I/Os related to database logging.

The monitor elements log_reads (number of log pages read) and log_writes (number of log pages written) return the amount of I/O activity related to database logging. You can use an operating system monitor tool to collect information about other disk I/O activity, then compare the two types of I/O activity.

With LDAP we highly recommend that you have your log files on another volume or drive other then the LDAP database instance. Use the following command to set the path to the DB2 log file directory:

```
UPDATE DATABASE CONFIGURATION FOR database_alias USING NEWLOGPATH path
```

Note: Be sure the database instance owner has write access to the specified path or the command fails.

SQL0993W: The new path to the log (newlogpath) in the database configuration file is not valid.

Explanation: The path to the log file is not valid for one of the following reasons:

► The path does not exist.

- A file with the correct name was found in the specified path, but it is not a log file for this database.

- The database manager instance id does not have permission to access the path or a log file.

The requested change is not made.

User Response: To change the path to the log file, submit a database configuration command with a valid value.

16.5.10 Configuration script

This is the bat file you can change it as needed on a Windows server:

```
# Restrictions: This must be run under db2 command window in the
# sqllib\bin directory. This script must be run under the context of the
# ldapdb2 user. It does not require write authority to the current
directory.
db2 get database configuration for ldapdb2 > c:\db2cfgbefor.out
db2 update database configuration for ldapdb2 using SORTHEAP 2500
db2 update database configuration for ldapdb2 using MAXLOCKS 80
db2 update database configuration for ldapdb2 using MINCOMMIT 1
db2 update database configuration for ldapdb2 using UTIL_HEAP_SZ 5000
db2 update database configuration for ldapdb2 using LOGFILSIZ 10000
db2 update database configuration for ldapdb2 using LOGPRIMARY 5
db2 update database configuration for ldapdb2 using LOGSECOND 50
db2 update database configuration for ldapdb2 using APPLHEAPSZ 2048
db2 connect to ldapdb2
# mem usage: 259.375 MB for servers that have 1 gig of memory
db2 alter bufferpool ibmdefaultbp size 49800
db2 alter bufferpool ldapbp size 2075
# mem usage: 1037.5 MB for servers that have 3 gig of memory
#db2 alter bufferpool ibmdefaultbp size 199200
#db2 alter bufferpool ldapbp size 8300
db2 terminate
db2 force applications all
sleep 1
db2stop
db2start
db2 connect to ldapdb2
db2 get database configuration for ldapdb2 > c:\db2cfgafter.out
db2 "select bpname,npages,pagesize from syscat.bufferpools"
db2 terminate
```

This file can be adapted to fit a Unix box with some minor changes. This will give you a report of how it was configured before the changes were made. (c:\db2cfgbefore.out) and also have a report on how it looks after the changes were made. (c:\db2cfgafter.out).

16.6 Directory size

It is very important that you due your homework and look at what size you directory is at now and what it will be in the future. Performance degrades significantly as your database size grows, it becomes necessary to readjust the sizes of the LDAP caches and DB2 buffer pools from time to time. There is also directory monitoring and maintiance that needs to be done from time to time depending on how offen you are adding or changing your directory and if you are using replication.

16.7 Optimization and organization

DB2 uses a sophisticated set of algorithms to optimize the access to data stored in a database. These algorithms depend upon many factors, including the organization of the data in the database, and the distribution of that data in each table. Distribution of data is represented by a set of statistics maintained by the database manager.

In addition, IBM Tivoli Directory Server creates a number of indexes for tables in the database. These indexes are used to minimize the data accessed in order to locate a particular row in a table.

In a read-only environment, the distribution of the data changes very little. However, with updates and additions to the database, it is not uncommon for the distribution of the data to change significantly. Similarly, it is quite possible for data in tables to become ordered in an inefficient manner.

To remedy these situations, DB2 provides tools to help optimize the access to data by updating the statistics and to reorganize the data within the tables of the database.

16.7.1 Optimization

Optimizing the database updates statistics related to the data tables, which improves performance and query speed. Optimize the database periodically or after heavy database updates (for example, after importing database entries). The Optimize database task in the IBM Tivoli Directory Server Configuration Tool uses the DB2 runstats command to update statistical information used by the query optimizer for all the LDAP tables.

> **Note:** The `reorgchk` command also updates statistics. If you are planning to do a `reorgchk`, optimizing the database is unnecessary.

Optimize the database using the Configuration Tool

To do this:

1. Start the Configuration Tool by typing `ldapxcfg` on the command line.

2. Click **Optimize database** on the left side of the window.

3. On the Optimize database window, click **Optimize**.

After a message displays indicating the database was successfully optimized, you must restart the server for the changes to take effect.

Optimize the database using the command line

Run the following command from the db2 command line. Each command needs to be all on one line:

```
DB2 RUNSTATS ON TABLE table_name AND DETAILED INDEXES ALL SHRLEVEL
REFERENCE
```

Run the following commands for more detailed lists of runstats that might improve performance (remember the each of these commands need to be on the same line):

```
DB2 RUNSTATS ON TABLE table_name WITH DISTRIBUTION AND DETAILED INDEXES ALL
SHRLEVEL REFERENCE
```

```
DB2 RUNSTATS ON TABLE ldapdb2.objectclass WITH DISTRIBUTION AND DETAILED
INDEXES ALL SHRLEVEL REFERENCE
```

Where *table_name* is the name of the table. You can use ALL for all tables.

16.7.2 reorgchk and reorg

Tuning the organization of the data in DB2 using the `reorgchk` and `reorg` commands is important for optimal performance. The `reorgchk` command updates statistical information to the DB2 optimizer to improve performance, and reports statistics on the organization of the database tables.

The `reorg` command, using the data generated by `reorgchk`, reorganizes table spaces to improve access performance and reorganizes indexes so that they are more efficiently clustered. The `reorgchk` and `reorg` commands can improve both search and update operation performance.

> **Note:** Tuning organizes the data on disk in a sorted order. Sorting the data on disk is beneficial only when accesses occur in a sorted order, which is not typically the case. For this reason, organizing the table data on disk typically yields little change in performance.

Performing a reorgchk

After appox 10,000 number of updates have been performed against DB2, table indexes can become sub-optimal and performance can degrade. The db2 reorgchk command is one of the most important and often over looked because it is not a one-time tuning item. As updates are performed on the DB2 database, the statistical information about the tables will not be up to date. The db2 reorgchk command update the important statistics that are used by the DB2 optimizer. To perform a DB2 reorgchk, do the following:

```
db2 connect to ldapdb2
db2 reorgchk update statistics on table all
```

Where *ldapdb2* is the name of your database.

To generate a reorgchk output file (recommended if you plan to run the reorg command) add the name of the file to the end of the command for a unix OS, for example:

```
db2 reorgchk update statistics on table all > /tmp/reorgchk.out
```

You can create a bat file for windows like this one and call it reorgchk.bat, you can change it as needed to fit your environment. Run this on the db2 command line:

```
db2 connect to ldapdb2
db2 reorgchk update statistics on table all > c:\reorgchk.out
db2 terminate
```

The output looks like the following:

```
E:\PROGRA~1\SQLLIB\BIN>reorgchk
E:\PROGRA~1\SQLLIB\BIN>db2 connect to ldapdb2
Database Connection Information
Database server        = DB2/NT 7.2.8
 SQL authorization ID   = ADMINIST...
 Local database alias   = LDAPDB2
E:\PROGRA~1\SQLLIB\BIN>db2 reorgchk update statistics on table all
1>c:\reorgchk.out
E:\PROGRA~1\SQLLIB\BIN>db2 terminate
DB20000I  The TERMINATE command completed successfully.
E:\PROGRA~1\SQLLIB\BIN>
```

Performing a reorg

After you have generated organizational information about the database using reorgchk, the next step in reorganization is finding the tables and indexes that need reorganizing and attempting to reorganize them. This can take a long time. The time it takes to perform the reorganization process increases as the DB2 database size increases.

In general, reorganizing a table takes more time than running statistics. Therefore, performance might be improved significantly by running statistics first.

Check the output of the reorgchk in the c:\ directory called reorgchk.out if you ran the script above.

If you look at the output and see "*" in the last column you should do a reorg of that table or index. To tell what is a table and what is an index just look on the output. The output has two sections. The first section talks to Tables the next section talks to Indexes.

Generally speaking, because most data in LDAP is accessed by index, reorganizing tables is usually not as beneficial as reorganizing indexes.

Reorgchk output showing a table that needs to be reorganized
The following is example output from the reorgchk that shows a table that needs to be reorganized:

```
SYSIBM    SYSINDEXES              282    90    17    29    184710  31 100  58 *-*
```

Reorgchk output showing an index that needs to be reorganized
The following is an example output from the reorgchk command that shows a table that needs to be reorganized:

```
Table: LDAPDB2.ACLPROP
LDAPDB2   ACLPROP_INDEX       63982   231    3     5   63982   100   87  101 *--
Table: LDAPDB2.DESCRIPTION
LDAPDB2   DESCRIPTION         32516   216    3     6   32516    99   51  175 --*
Table: LDAPDB2.USER_BOBCS_EMPLID
LDAPDB2   RUSER_BOBCS_EMPLID  19430   148    3    14   19430     2   70  129 *-*
```

Procedure to perform a reorganization using the reorg command
Follow these steps to perform a reorganization using the reorg command.

Open up a db2 command window.

Enter the following commands using the examples from above output:

```
db2 connect to ldapdb2
db2 reorg table SYSIBM.SYSINDEXES
db2 reorg table LDAPDB2.ACLPROP index LDAPDB2.ACLPROP_INDEX
db2 reorg table LDAPDB2.DESCRIPTION index LDAPDB2.DESCRIPTION
db2 reorg table LDAPDB2.USER_BOBCS_EMPLID index LDAPDB2.RUSER_BOBCS_EMPLID
```

The output looks like this:

```
E:\PROGRA~1\SQLLIB\BIN>db2 connect to ldapdb2
Database Connection Information
Database server        = DB2/NT 7.2.8
```

```
 SQL authorization ID   = ADMINIST...
 Local database alias   = LDAPDB2
E:\PROGRA~1\SQLLIB\BIN>db2 reorgchk update statistics on table all >
e:\migration\reorgchk.out
E:\PROGRA~1\SQLLIB\BIN>db2 reorg table SYSIBM.SYSINDEXES
DB20000I  The REORG TABLE command completed successfully
E:\PROGRA~1\SQLLIB\BIN>db2 reorg table LDAPDB2.ACLPROP index
LDAPDB2.ACLPROP_INDEX
DB20000I  The REORG TABLE command completed successfully.
E:\PROGRA~1\SQLLIB\BIN>db2 reorg table LDAPDB2.DESCRIPTION index
LDAPDB2.DESCRIPTION
DB20000I  The REORG TABLE command completed successfully
E:\PROGRA~1\SQLLIB\BIN>db2 reorg table LDAPDB2.USER_BOBCS_EMPLID index
LDAPDB2.RUSER_BOBCS_EMPLID
DB20000I  The REORG TABLE command completed successfully.
```

Keep in mind that reorgchk needs to be run periodically. For example, reorgchk might need to be run after a large number of updates have been performed.

Note: LDAP tools such as ldapadd, ldif2db, and bulkload can potentially do large numbers of updates that require a reorgchk. The performance of the database should be monitored and a reorgchk performed when performance starts to degrade.

A reorgchk must be performed on all LDAP replicas because each replica uses a separate database. The LDAP replication process does not include the propagation of database optimizations.

After you reorg all the ones that needed to be reorged you need to run reorgchk again. The output from reorgchk can then be used to determine whether the reorganization worked and whether it introduced other tables and indexes that need reorganizing.

Some guidelines for performing a reorganization are:

► If the number on the column that has an asterisk is close to the recommended value described in the header of each section and one reorganization attempt has already been done, you can probably skip a reorganization on that table or index.

► In the table LDAPDB2.LDAP_ENTRY there exists a LDAP_ENTRY_TRUNC index and a SYSIBM.SQL index. Preference should be given to the SYSIBM.SQL index if attempts to reorganize them seem to alternate between one or the other needing reorganization.

► Reorganize all the attributes that you want to use in searches. In most cases you will want to reorganize to the forward index, but in cases with searches beginning with '*', reorganize to the reverse index. For example:

```
Table: LDAPDB2.SECUUID
LDAPDB2 RSECUUID <. This is a reverse index
LDAPDB2 SECUUID <. This is a forward index
LDAPDB2 SECUUIDI <. This is an update index
```

16.7.3 Indexes

Indexing results in a considerable reduction in the amount of time it takes to locate requested data. For this reason, it can be very beneficial from a performance standpoint to index all attributes used in searches. You can use the audit log to find out what attributes are being used in searches. Then you will need to use the following to find out if that attribute is indexed.

Use the following DB2 commands to verify that a particular index is defined. In the following example, the index being checked is for the attribute principalName:

```
db2 connect to database_name
db2 list tables for all | grep -i principalName
db2 describe indexes for table database_name.principalName
```

Where *database_name* is the name of your database.

If the second command fails or the last command does not return three entries, the index is not properly defined. The last command should return results similar to Example 16-2.

Example 16-2   Showing defined indexes

IndexScheme of Columns	Index Name	Unique Rule	Number
LDAPDB2	PRINCIPALNAME[	D	1
LDAPDB2	PRINCIPALNAME	D	2
LDAPDB2	RPRINCIPALNAME	D	2

```
            3 record(s) selected
```

To have IBM Tivoli Directory Server create an index for an attribute the next time the server is started, do one of the following:

► To create an index using the Web Administration Tool:

a. Expand Schema management in the navigation area, and click **Manage attributes**.

b. Click **Edit attribute**.

c. On the IBM extensions tab, select the Equality check box under Indexing rules, and you can check for a number of types of indexes from this screen.

► To create an index from the command line, issue the following command:

```
ldapmodify -f /ldap/etc/addindex.ldif
```

The addindex.ldif file should look like this:

> **Note:** Make sure that there is a "-" between each attribute that is being changed on the same DN, and there is no space between each line of the same DN. Also for any objectclass, attributetypes entries they need to be all on the same line in the document. There should only be one space between each DN.

```
dn: cn=schema
changetype: modify
replace: attributetypes
attributetypes: ( 1.3.18.0.2.4.318 NAME ( 'principalName' 'principal' )
DESC
'A naming attribute that may be used to identify eUser object entries.'
EQUALITY
1.3.6.1.4.1.1466.109.114.2 ORDERING 2.5.13.3 SUBSTR 2.5.13.4 SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 USAGE userApplications )
-
replace: ibmattributetypes
ibmattributetypes: ( 1.3.18.0.2.4.318 DBNAME( 'principalName'
'principalName' )
ACCESS-CLASS normal LENGTH 256 EQUALITY ORDERING SUBSTR APPROX )
```

16.7.4 Distributing the database across multiple physical disks

As the database grows, it becomes necessary and desirable to distribute the database across multiple physical disk drives. You can achieve better performance by spreading entries across multiple disks. In terms of performance, one 20 GB disk is not as good as two 10 GB disks. The following sections describe how to configure DB2 to distribute the ldapdb2 database across multiple disks.

IBM Directory tablespaces

When IBM Directory creates a database for the directory, it uses the db2 create database command to create a database named ldapdb2. IBM Directory Server creates this database with four System Managed Space (SMS) tablespaces. You

can view the tablespaces by using the following DB2 commands run under the context of the DB2 instance owner, typically the ldapdb2 user:

```
db2 connect to ldapdb2
db2 list tablespaces
```

The following examples show UNIX tablespace output for IBM Directory:

```
Tablespaces for Current Database
Tablespace ID = 0
Name = SYSCATSPACE
Type = System managed space
Contents = Any data
State = 0x0000
Detailed explanation:
Normal
Tablespace ID = 1
Name = TEMPSPACE1
Type = System managed space
Contents = Temporary data
State = 0x0000
Detailed explanation:
Normal
Tablespace ID = 2
Name = USERSPACE1
Type = System managed space
Contents = Any data
State = 0x0000
Detailed explanation:
Normal
Tablespace ID = 3
Name = LDAPSPACE1
Type = System managed space
Contents = Any data
State = 0x0000
Detailed explanation:
Normal
```

IBM Directory is stored in the user tablespace (USERSPACE1) and in the IBM Directory tablespace (LDAPSPACE). By default, there is only one container or directory for the user tablespace. To view the details about the user tablespace, enter a DB2 command similar to the following:

```
db2 list tablespace containers for 2
```

Example output is as follows:

```
Tablespace Containers for Tablespace 2
Container ID = 0
Name = /ldapdb2/NODE0000/SQL00001/SQLT0002.0
Type = Path
```

The container or directory that DB2 uses for tablespace 2 is /ldapdb2/SQL00001/SQLT0002.0. It contains some of the ldapdb2 database tables. Tablespace 3 contains the remainder of the database tables, the biggest of which is the ldap_entry table. The ldap_entry table contains the majority of the IBM Directory data.

16.7.5 Create file systems and directories on the target disks

The first step in distributing the DB2 database across multiple disk drives is to create and format the file systems and directories on the physical disks that the database is to be distributed among.

Guidelines are as follows:

► Because DB2 distributes the database equally across all directories, it is a good idea to make all of the file systems, directories, or both, the same size.

► All directories to be used for the DB2 database must be completely empty. AIX and Solaris systems create a lost+found directory at the root of any file system. Instead of deleting the lost+found directory, create a subdirectory at the root of each file system to be used for distributing the database. For example, create a subdirectory named c in each filesystem where the DB2 database is to be stored.

► Create two additional directories under the c directory: one for holding tablespace 2 and the other for tablespace 3. For example, these directories might be named 2 and 3. Then specify these directories on the set tablespace commands as discussed in "Perform a redirected restore of the database."

The DB2 instance user must have Write permission on the created directories. For AIX and Solaris systems, the following command gives the proper permissions:

```
chown ldapdb2 directory_name
```

The following are platform-specific guidelines:

► For the AIX operating system, create the file system with the Large File Enabled option. This option is one of the options on the Add a Journaled File System smit menu.

► For AIX and Solaris systems, set the file size limit to unlimited or to a size large enough to allow for the creation of a file as large as the file system. On AIX systems, the /etc/security/limits file controls system limits and -1 means unlimited. On Solaris systems, the ulimit command controls system limits.

16.7.6 Backing up the existing database

To back up the existing database, follow these steps:

1. Stop the IBM Directory Server process (slapd or ibmslapd).

2. To close all DB2 connections, enter the following:

```
db2 force applications all
db2 list applications
```

A message similar to the following is displayed:

SQL1611W No data was returned by Database System Monitor.

3. To initiate the backup process, enter the following:

```
db2 backup db ldapdb2 to [file system | tape device]
```

When the database has been backed up successfully, a message similar to the following is displayed:

```
Backup successful. The timestamp for this backup image is : 20000420204056
```

Note: Ensure that the backup process was successful before proceeding. The next step destroys the existing database in order to recreate it. If the backup was not successful, the existing database is lost. You can verify the success of the backup by restoring to a separate system.

16.7.7 Perform a redirected restore of the database

A DB2 redirected restore restores the specified database tablespace to multiple containers or directories. In the following example, assume that the following directories for containing tablespace 2 were created, are empty, and have the correct permissions to allow write access by the DB2 instance owner, typically the ldapdb2 user:

```
/disks/1/c/2
/disks/2/c/2
/disks/3/c/2
/disks/4/c/2
/disks/5/c/2
```

In the following example, assume the following directories for tablespace 3 were created:

```
/disks/1/c/3
/disks/2/c/3
/disks/3/c/3
/disks/4/c/3
/disks/5/c/3
```

Redirected restore

To do a redirected restore:

1. To start the DB2 restore process, enter the following:

   ```
   db2 restore db ldapdb2 from [location of backup] replace existing redirect
   ```

   Messages similar to the following are displayed:

   SQL2539W Warning! Restoring to an existing database that is the same as the
   backup image database. The database files will be deleted.
   SQL1277N Restore has detected that one or more tablespace containers are
   inAccessible, or has set their state to 'storage must be defined'.
   DB20000I The RESTORE DATABASE command completed successfully.

2. To define the containers for tablespace 2 and for tablespace 3, enter the
   following:

   ```
   db2 "set tablespace containers for 2 using (path \
   '/disks/1/c/2', path '/disks/2/c/2', path '/disks/3/c/2', \
   path '/disks/4/c/2', path '/disks/5/c/2')"

   db2 "set tablespace containers for 3 using (path \
   '/disks/1/c/3', path '/disks/2/c/3', path '/disks/3/c/3', \
   path '/disks/4/c/3', path '/disks/5/c/3')"
   ```

   > **Note:** If many containers are defined, these commands can become so
   > long as to not fit within the limits of a shell command. In this case, you can
   > put the command in a file and run within the current shell using the dot
   > notation. For example, assume that the commands are in a file named
   > set_containers.sh. The following command runs it in the current shell:
   >
   > ```
   > . set_containers.sh
   > ```

   After completion of the DB2 set tablespace command, a message similar to
   the following is displayed:

   DB20000I The SET TABLESPACE CONTAINERS command completed successfully.

   If you receive the following message:

   SQL0298N Bad container path. SQLSTATE=428B2

> **Note:** A newly created file system on AIX and Solaris contains a directory
> named lost+found. You should create a directory at the same level as
> lost+found to hold the tablespace and then reissue the set tablespace
> command. If you experience problems, see the DB2 documentation. The
> following files might also be of interest:
>
> ```
> ldapdb2_home_dir /sqllib/Readme/en_US/Release.Notes
> ldapdb2_home_dir /sqllib/db2dump/db2diag.log
> ```
>
> The db2diag.log file contains some fairly low-level details that can be
> difficult to interpret.

This indicates that one of the containers is not empty, or that Write permission
is not enabled for the DB2 instance owner, typically the ldapdb2 user.

3. Continue the restore to new tablespace containers. This step takes the most
   time to complete. The time varies depending on the size of the directory. To
   continue the restore to the new tablespace containers, enter the following:

```
db2 restore db ldapdb2 continue
```

If problems occur with the redirected restore and you want to restart the restore
process, it might be necessary to enter the following command first:

```
db2 restore db ldapdb2 abort
```

16.8 DB2 backup and restore

The fastest way to back up and restore the database is to use DB2 backup and
restore commands. LDAP alternatives, such as db2ldif and ldif2db, are generally
much slower in comparison.

The only disadvantage to using the db2 backup and db2 restore commands is
that the backed-up database cannot be restored across dissimilar hardware
platforms. For example, you cannot back up an AIX database and restore the
database to a Solaris system. An alternative to the db2 backup and db2 restore
commands is an LDAP Information File (LDIF) export and import. These
commands work across dissimilar hardware platforms, but the process is slower.
For more information about the use of these commands, see the DB2
documentation.

An important advantage of using db2 backup and db2 restore commands is the
preservation of DB2 configuration parameters and db2 reorgchk database
optimizations in the backed-up database. The restored database has the same

performance tuning tasks as the backed-up database. This is not the case with LDAP db2ldif and ldif2db.

Be aware that if you restore over an existing database, any performance tuning tasks on that existing database are lost. Check all DB2 configuration parameters after performing a restore. Also, if you do not know whether a db2 reorgchk was performed before the database was backed up, run db2 reorgchk after the restore. The DB2 commands to perform backup and restore operations are as follows:

```
db2 force applications all
db2 backup db ldapdb2 to directory_or_device
db2 restore db ldapdb2 from directory_or_device replace existing
```

Where *directory_or_device* is the name of a directory or device where the backup is stored.

The most common error that occurs on a restore is a file permission error. Following are some reasons why this error might occur:

The DB2 instance owner does not have permission to access the specified directory and file. One way to solve this is to change directory and file ownership to the DB2 instance owner. For example, enter the following:

```
chown ldapdb2 fil_or_dev
```

The backed-up database is distributed across multiple directories, and those directories do not exist on the target system of the restore. Distributing the database across multiple directories is accomplished with a redirected restore. To solve this problem, either create the same directories on the target system or perform a redirected restore to specify the proper directories on the new system. If creating the same directories, ensure that the owner of the directories is the DB2 instance owner typically the ldapdb2 user.

Backup and restore operations are required to initially synchronize an LDAP replica with an LDAP master or whenever the master and replica get out of sync. A replica can get out of sync if it is not defined to the master. In this case, the master does not know about the replica and does not save updates on a propagation queue for that replica.

If a newly configured master LDAP directory is to be loaded with initial data, you can use bulk-loading utilities to speed up the process. This is another case in which the replica is not informed of updates and a manual backup and restore is required to get the replica synchronized with the master.

16.9 Concurrent updates on Symmetric Multi-Processor systems

Better update performance, particularly on Symmetric Multi-Processor (SMP) systems, is typically achieved by making updates concurrently (for example, with multiple concurrent update clients). In some cases, update performance might not improve with concurrent updates, specifically when the LDAP propagation queue grows very large. This can happen when the LDAP master server does updates faster than those updates can be propagated to the LDAP replica servers. Because propagation is done serially, concurrent updates on the master are likely to result in a growth of the propagation queue. Some testing is required in a master/replica environment to determine how much performance improvement, if any, comes from concurrent updates.

16.10 AIX operating system tuning

In this section we discuss AIX operating system tuning.

16.10.1 Enabling large files

The underlying AIX operating system files that hold the contents of a large directory can grow beyond the default size limits imposed by the AIX operating system. If the size limits are reached, the directory ceases to function correctly. The following steps make it possible for files to grow beyond default limits on an AIX operating system:

1. When you create the file systems that are expected to hold the directory's underlying files, you should create them as Large File Enabled Journaled File Systems. The file system containing the DB2 instance's home directory, and, if bulkload is used, the file system containing the bulkload temporary directory, are file systems that can be created this way.

2. Set the soft file size limit for the root, ldap, and the DB2 instance owner users to -1. A soft file size limit of -1 for a user specifies the maximum file size for that user as unlimited. The soft file size limit can be changed using the smitty chuser command. Each user must log off and log back in for the new soft file size limit to take effect. You must also restart DB2.

Setting MALLOCMULTIHEAP

The MALLOCMULTIHEAP environment variable can improve LDAP performance on symmetric multi-processor (SMP) systems. To set this variable, run the following command just before starting ibmslapd, in the same environment where you will start ibmslapd:

```
export MALLOCMULTIHEAP=1
```

The disadvantage of using MALLOCMULTIHEAP is increased memory usage. It might take less memory, yet have less of a performance benefit, if the variable is set as follows:

```
export MALLOCMULTIHEAP=heaps: numprocs+1
```

Where *numprocs* is the number of processors in the multiprocessor system.

You can find more information about MALLOCMULTIHEAP in the AIX documentation.

Setting MALLOCTYPE

Set the MALLOCTYPE environment variable as follows, according to the version of AIX you are running:

▶ AIX 5.1

   Set MALLOCTYPE as follows:

   ```
   export MALLOCTYPE=buckets
   ```

▶ AIX 5.2

   Set MALLOCTYPE to the default. If you have already set MALLOCTYPE to another value, you can set it to the default by typing the following:

   ```
   export MALLOCTYPE=null
   ```

You can find more information about MALLOCTYPE in the AIX documentation.

Setting other environment variables

You might experience better performance by setting the AIXTHREAD_SCOPE and NODISCLAIM environment as shown in the following commands. Check the AIX documentation to see if these settings might be right for your installation.

To set AIXTHREAD_SCOPE, use the following command:

```
export AIXTHREAD_SCOPE=S
```

To set NODISCLAIM, use the following command:

```
export NODISCLAIM=TRUE
```

16.10.2 Tuning process memory size limits

On UNIX platforms, some of LDAP performance tuning tasks in this document result in process sizes that exceed the operating system default limits. This section describes how to increase the operating system limits so that the affected processes do not run out of memory. When a process runs out of memory, the process often ends. In some cases, it leaves a core dump file, an error message, or an error log entry. On AIX systems, the system error log might indicate that the

process ended due to memory allocation failure. Use the errpt –a | more command to display the error log.

Increasing the operating system process memory size limits

On UNIX systems, each user can either inherit resource limits from the root user or have specific limits defined. The most useful setting to use for the process size limits is unlimited. That way, the system process size limits are defined to allow the maximum process growth.

On AIX systems, the process size limits are defined in the /etc/security/limits file. A value of -1 indicates that there is either no limit or that it is unlimited. The names of the limits to increase are data and rss. For changes to the /etc/security/limits file to take effect, the user must log out of the current login session and log back in. On AIX, some limits may apply to the root user.

On Solaris systems, the process size limits are defined by the ulimit command. You can specify a value of unlimited on the command. The names of the limits to increase are data and vmemory. By default on Solaris systems, the root user has unlimited access to these resources (for example, unlimited).

When setting resource limits for a process, it is important to know that the limits that apply are those that are in effect for the parent process and not the limits for the user under which the process runs.

16.10.3 AIX-specific process size limits

On AIX systems, the number of data segments that a process is allowed to use also limits the process memory size. The default number of data segments is 1. The size of a data segment is 256 MB. Data segments are shared for both data and stack. The maximum number of data segments a process can use is 8.

Setting the maximum number of AIX data segments On AIX, the number of segments that a process can use for data is controlled by the LDR_CNTRL environment variable. It is defined in the parent process of the process that is to be affected. For example, the following example defines one additional data segment:

```
export LDR_CNTRL=MAXDATA=0x10000000
start_process
unset LDR_CNTRL
```

It is a good idea to unset the LDR_CNTRL environment variable, so that it does not unintentionally affect other processes.

Unlike other environment variables for the IBM Directory Server process (slapd or ibmslapd), the LDR_CNTRL environment variable cannot be set as a

front-end variable in the slapd32.conf or ibmslapd.conf configuration file. It must be set as an environment variable.

16.10.4 AIX data segments and LDAP process DB2 connections

On AIX, process segments are used for increasing a process memory size and for shared memory. A segment can be used for one or the other of these purposes, but not for both. When possible, the IBM Directory Server uses shared memory to communicate between its server process (slapd or ibmslapd) and DB2 processes. Each shared segment used in this way is a connection to the DB2 database. If there are not enough process segments to satisfy the number of DB2 connections defined in the IBM Directory Server configuration file (slapd32.conf or imbslapd.conf), the remaining connections are satisfied by using local TCP/IP sockets. For this reason, there is no conflict between increasing the process memory size of the IBM Directory Server process and increasing the number of DB2 connections defined for the IBM Directory Server to use.

16.10.5 Verifying process data segment usage

If the perfagent.tools are installed, /usr/bin/svmon -P pid shows the memory usage of a process. In the output, identify the segments labeled shmat/mmap. Segments with an Inuse column of zero (0) are for data segments that are available for process growth. Segments with an Inuse column greater than 1 are for data segments in which the process has already grown. Segments with an Inuse column of 1 are usually found in the slapd or the ibmslapd process and represent the shared memory segments being used for DB2 connections.

16.11 Adding memory after installation on Solaris systems

Memory added to a computer after the installation of a Solaris operating system does not automatically improve performance. To take advantage of added memory, you must:

1. Update the shared memory (shmem) value in the /etc/system file:

```
set shmsys:shminfo_shmmax = physical_memory
```

Where *physical_memory* is the size on of the physical memory on the computer in bytes. You must restart the computer for the new settings to take effect.

2. From the command line, set the ulimit values for increasing process memory and file size to unlimited:

```
ulimit -d unlimited
ulimit -v unlimited
```

```
ulimit -f unlimited
```

16.12 SLAPD_OCHANDLERS variable on Windows

On Windows, if you have clients that are generating many connections to the server and the connections are being refused, set the SLAPD_OCHANDLERS environment variable to 5 before starting the server.

16.13 IBM Directory Change and Audit Log

In this section we discuss the IBM Directory Change and Audit Log.

16.13.1 When to configure the LDAP change log

IBM Tivoli Directory Server has a function called change log that results in a significantly slower LDAP update performance. The change log function should be configured only if needed.

The change log function causes all updates to LDAP to be recorded in a separate change log DB2 database (that is, a different database from the one used to hold the LDAP server Directory Information Tree). The change log database can be used by other applications to query and track LDAP updates. The change log function is disabled by default.

One way to check for existence of the change log function is to look for the suffix CN=CHANGELOG. by issuing the following command:

```
ldapsearch -D cn=root -w <dir admin password> -b "" "objectclass=*
```

If it exists, the change log function is enabled. To disable this after it is configured, run the following commands with the directory shutdown. If you try to do this with the directory up you will get the following message:

```
Unable to unconfigure database while IBM Tivoli Directory Server is
running.
```

For IBM Directory 4.1 and earlier:

```
ldapucfg -g
```

For IBM Directory 5.1 and later:

```
ldapxcfg
```

and click Configure/unconfigure changelog and uncheck box on the right and then click update and it will unconfigure the changelog. You will get a message

stating that disabling changelog will destroy the data currently in the changelog
database. Do you still want to continue? If you want to save the changelog you
will need to copy it off before you answer this yes, if not it will be deleted.

16.13.2 When to configure the LDAP audit log

The audit log shows what searches are being performed and the parameters
used in each search. The audit log also shows when a client binds and unbinds
from the directory. Observing these measurements allows you to identify LDAP
operations that take a long time to complete. Depending on what options you
have turned on to monitor this can cause a significantly slow down of many
aspects of the IBM Directory Server performance. It is recommended that all
audit logging features be turned off unless your are troubleshooting searches or
need to record for security reasons. Do the following:

```
ldapsearch -D cn=root -w <passwd> -b "cn=audit,cn=localhost" objectclass=*
```

With 4.1 and earlier you will see the following included in the output:

```
cn=audit,cn=localhost
objectclass=ibm-auditConfig
objectclass=top
cn=audit
ibm-auditlog=C:\Program Files\IBM\LDAP\var\audit
ibm-audit=false
ibm-auditfailedoponly=true
ibm-auditbind=true
ibm-auditunbind=true
ibm-auditsearch=false
ibm-auditadd=false
ibm-auditmodify=false
ibm-auditdelete=false
ibm-auditmodifydn=false
ibm-auditextopevent=false
```

With 5.1 and later you will see the following included in the output:

```
CN=AUDIT,CN=LOCALHOST
objectclass=ibm-auditConfig
objectclass=ibm-slapdConfigEntry
objectclass=top
cn=audit
ibm-auditLog=C:\Program Files\IBM\LDAP\var\audit.log
ibm-auditVersion=2
ibm-audit=false
ibm-auditFailedOPonly=true
ibm-auditBind=true
ibm-auditUnbind=true
ibm-auditSearch=false
```

```
ibm-auditAdd=false
ibm-auditModify=false
ibm-auditDelete=false
ibm-auditModifyDN=false
ibm-auditExtOPEvent=false
ibm-auditExtOp=false
```

You can use the Web admin tool for each of the versions of the Directory to set these settings to what you want.

16.14 Hardware tuning

In this section we discuss hardware tuning.

16.14.1 Disk speed improvements

With millions of entries in LDAP server, it can become impossible to cache all of them in memory. Even if a smaller directory size is cacheable, update operations must go to disk. The speed of disk operations is important. Here are some considerations for helping to improve disk drive performance:

▶ Use fast disk drives.

▶ Use a hardware write cache.

▶ Spread data across multiple disk drives.

▶ Spread the disk drives across multiple I/O controllers.

▶ Put log files and data on separate physical disk drives.

16.15 Monitoring performance

The `ldapsearch` command can be used to monitor performance, as shown in the following sections.

16.15.1 ldapsearch with "cn=monitor"

The following ldapsearch command uses "cn=monitor".

```
ldapsearch -h ldap_host -s base -b cn=monitor objectclass=*
```

Where *ldap_host* is the name of the LDAP host.

With 5.1 and earlier

We had these limited outputs to the cn=monitor command. The monitor search returns the following attributes of the server:

- ► version: Version of the LDAP server
- ► totalconnections: Total number of connections to the server
- ► currentconnections: Total number of current connections
- ► maxconnections: Configured maximum number of connections
- ► writewaiter: Number of threads waiting to write
- ► readwaiters: Number of threads waiting to read
- ► opsinitiated: Operations initiated against the server
- ► opscompleted: Number of operations completed
- ► entriessent: Number of entries sent from the server
- ► searchesrequested: Number of searches requested
- ► searchescompleted: Number of searches completed
- ► filter_cache_size: Configured maximum size of the filter cache
- ► filter_cache_current: Current size of the filter cache
- ► filter_cache_click: Number of searches that have click the filter cache
- ► filter_cache_miss: Number of searches that have missed the filter cache
- ► entry_cache_size: Configured maximum size of the entry cache
- ► entry_cache_current: Current size of the entry cache
- ► entry_cache_click: Number of entries returned from entry cache
- ► entry_cache_miss: Number of entries returned not from entry cache
- ► currenttime: Current time of the search
- ► starttime: Start time of the server
- ► en_currentregs: Number of events currently registered
- ► en_notificationssent: Number of event notifications sent

With 5.2 and later

A number of upgrades to the cn=monitor command allows it to pull out more data to better monitor how the LDAP is doing. The monitor search returns some of the following attributes of the server:

- ► cn=monitor
- ► version=IBM Tivoli Directory, Version 5.2
- ► total connections: The total number of connections since the server was started
- ► current connections: The number of active connections
- ► maxconnections: The maximum number of active connections allowed
- ► writewaiters: The number of threads sending data back to the client
- ► readwaiters: The number of threads reading data from the client
- ► opsinitiated: The number of initiated requests since the server was started

- ► livethreads: The number of worker threads being used by the server

- ► opscompleted: The number of completed requests since the server was started

- ► entriessent: The number of entries sent by the server since the server was started.

- ► searchesrequested: The number of initiated searches since the server was started.

- ► searchescompleted: The number of completed searches since the server was started.

- ► filter_cache_size: The maximum number of filters allowed in the cache.

- ► filter_cache_current: The number of filters currently in the cache.

- ► filter_cache_click: The number of filters retrieved from the cache rather than being resolved in DB2.

- ► filter_cache_miss: The number of filters that were not found in the cache that then needed to be resolved by DB2.

- ► filter_cache_bypass_limit: Search filters that return more entries than this limit are not cached.

- ► entry_cache_size: The maximum number of entries allowed in the cache.

- ► entry_cache_current: The number of entries currently in the cache.

- ► entry_cache_click: The number of entries that were retrieved from the cache.

- ► entry_cache_miss: The number of entries that were not found in the cache that then needed to be retrieved from DB2.

- ► acl_cache: A Boolean value indicating that the ACL cache is active (TRUE) or inactive (FALSE).

- ► acl_cache_size: The maximum number of entries in the ACL cache.

- ► currenttime: The current time on the server. The current time is in the format: year month day hour:minutes:seconds GMT.

  Note: If expressed in local time the format is day month date hour:minutes:seconds timezone year.

- ► starttime: The time the server was started. The start time is in the format: year month day hour:minutes:seconds GMT.

  Note: If expressed in local time the format is day month date hour:minutes:seconds timezone year.

- ► en_currentregs: The current number of client registrations for event notification.
- ► en_notificationssent: The total number of event notifications sent to clients since the server was started.

The following attributes are for operation counts:

- ► bindsrequested: The number of bind operations requested since the server was started
- ► bindscompleted: The number of bind operations completed since the server was started
- ► unbindsrequested: The number of unbind operations requested since the server was started
- ► unbindscompleted: The number of unbind operations completed since the server was started
- ► addsrequested: The number of add operations requested since the server was started
- ► addscompleted: The number of add operations completed since the server was started
- ► deletesrequested: The number of delete operations requested since the server was started
- ► deletescompleted: The number of delete operations completed since the server was started
- ► modrdnsrequested: The number of modify RDN operations requested since the server was started
- ► modrdnscompleted: The number of modify RDN operations completed since the server was started
- ► modifiesrequested: The number of modify operations requested since the server was started
- ► modifiescompleted: The number of modify operations completed since the server was started
- ► comparesrequested: The number of compare operations requested since the server was started
- ► comparescompleted: The number of compare operations completed since the server was started
- ► abandonsrequested: The number of abandon operations requested since the server was started
- ► abandonscompleted: The number of abandon operations completed since the server was started

- extopsrequested: The number of extended operations requested since the server was started

- extopscompleted: The number of extended operations completed since the server was started

- unknownopsrequested: The number of unknown operations requested since the server was started

- unknownopscompleted: The number of unknown operations completed since the server was started

The following attributes are for server logging counts:

- slapderrorlog_messages: The number of server error messages recorded since the server was started or since a reset was performed

- slapdclierrors_messages: The number of DB2 error messages recorded since the server was started or since a reset was performed

- auditlog_messages: The number of audit messages recorded since the server was started or since a reset was performed

- auditlog_failedop_messages: The number of failed operation messages recorded since the server was started or since a reset was performed

The following attributes are for connection type counts:

- total_ssl_connections: The total number of SSL connections since the server was started

- total_tls_connections: The total number of TLS connections since the server was started

The following attributes are for tracing:

- trace_enabled : The current trace value for the server. TRUE, if collecting trace data, FALSE, if not collecting trace data.

- trace_message_level: The current ldap_debug value for the server. The value is in hexadecimal form, for example:

  ```
  0x0=0
  0xffff=65535
  ```

- trace_message_log: The current LDAP_DEBUG_FILE environment variable setting for the server.

The following attributes are for denial of service prevention:

- available_workers: The number of worker threads available for work.

- current_workqueue_size: The current depth of the work queue.

- ► largest_workqueue_size: The largest size that the work queue has ever reached.

- ► idle_connections_closed: The number of idle connections closed by the Automatic Connection Cleaner.

- ► auto_connection_cleaner_run: The number of times that the Automatic Connection Cleaner has run.

- ► emergency_thread_running: The indicator of whether the emergency thread is running.

- ► totaltimes_emergency_thread_run: The number of times the emergency thread has been activated.

- ► lasttime_emergency_thread_run: The last time the emergency thread was activated.

The following attribute has been added for alias dereference processing:

- ► bypass_deref_aliases: The server runtime value that indicates if alias processing can be bypassed. It displays TRUE if no alias object exists in the directory, and FALSE if at least one alias object exists in the directory.

The following attributes are for the attribute cache:

- ► cached_attribute_total_size: The amount of memory used by the directory attribute cache, in kilobytes. This number includes additional memory used to manage the cache that is not charged to the individual attribute caches. Consequently, this total is larger than the sum of the memory used by all the individual attribute caches.

- ► cached_attribute_configured_size: The maximum amount of memory, in kilobytes, assigned to the directory attribute cache.

- ► cached_attribute_click: The number of times the attribute has been used in a filter that could be processed by the attribute cache. The value is reported as follows:

  ```
  cached_attribute_click=attrname:#####
  ```

- ► cached_attribute_size: The amount of memory used for this attribute in the attribute cache. This value is reported in kilobytes as follows:

  ```
  cached_attribute_size=attrname:######
  ```

- ► cached_attribute_candidate_click: A list of up to ten most frequently used noncached attributes that have been used in a filter that could have been processed by the directory attribute cache if all of the attributes used in the filter had been cached. The value is reported as follows:

  ```
  cached_attribute_candidate_click=attrname:#####
  ```

You can use this list to help you decide which attributes you want to cache. Typically, you want to put a limited number of attributes into the attribute cache because of memory constraints.

16.15.2 Monitor examples

The following sections show examples of using values returned by the ldapsearch command with cn=monitor to calculate the throughput of the server and the number of add operations completed on the server in a certain timeframe.

Throughput example

The following example shows how to calculate the throughput of the server by monitoring the server statistic called opscompleted, which is the number of operations completed since the LDAP server started.

Suppose the values for the opscompleted attribute obtained by issuing two ldapsearch commands to monitor the performance statistics, one at time t1 and the other at a later time t2, were opscompleted (t1) and opscompleted(t2.) The average throughput at the server during the interval between t1 and t2 can be calculated as:

```
(opscompleted(t2) - opscompleted(t1) - 3)/(t2 -t1)
```

Three is subtracted to account for the number of operations performed by the ldapsearch command itself.

Workload example

The monitor attributes can be used to characterize the workload, similar to the throughput example but split out by type of operation.

For example, you can calculate the number of add operations that were completed in a certain amount of time.

Suppose the values for the addscompleted attribute obtained by issuing two **ldapsearch** commands to monitor the performance statistics, one at time t1 and the other at a later time t2, were addscompleted (t1) and addscompleted(t2.) The number of add operations completed on the server during the interval between t1 and t2 can be calculated as:

```
(addscompleted(t2) - addscompleted(t1) - 3)/(t2 -t1)
```

Three is subtracted to account for the number of operations performed by the **ldapsearch** command itself.

Similar calculations can be done for other operations, such as searchescompleted, bindscompleted, deletescompleted, and modifiescompleted.

ldapsearch with "cn=workers,cn=monitor"

You can run a search using "cn=workers,cn=monitor" to get information about what worker threads are doing and when they started doing it.

```
ldapsearch -D <adminDN> -w <adminpw> -b cn=workers,cn=monitor -s base
objectclass=*
```

This information is most useful when a server is performing poorly or not functioning as expected. It should be used only when needed to give insight into what the server is currently doing or not doing.

The "cn=workers, cn=monitor" search returns detailed activity information only if auditing is turned on. If auditing is not on, "cn=workers, cn=monitor" returns only thread information for each of the workers.

> **Attention:** The cn=workers,cn=monitor search suspends all server activity until it is completed. For this reason, a warning should be issued from any application before issuing this feature. The response time for this command will increase as the number of server connections and active workers increase.

For more information, see the IBM Tivoli Directory Server Version 5.2 Administration Guide.

ldapsearch with "cn=connections,cn=monitor"

You can run a search using "cn=connections,cn=monitor" to get information about server connections:

```
ldapsearch -D<adminDN> -w <adminPW> -h <servername> -p <portnumber> -b
cn=connections,cn=monitor -s base objectclass=*
```

This command returns information in the following format:

```
cn=connections,cn=monitor
connection=1632 : 9.41.21.31 : 2002-10-05 19:18:21 GMT : 1 : 1 : CN=ADMIN : :
connection=1487 : 127.0.0.1 : 2002-10-05 19:17:01 GMT : 1 : 1 : CN=ADMIN : :
```

> **Note:** If appropriate, an SSL or a TLS indicator is added on each connection.

ldapsearch with "cn=changelog,cn=monitor"

You can run a search using "cn=changelog,cn=monitor" to obtain information about the changelog attribute cache. (See 16.13.1, "When to configure the LDAP change log" on page 533, for information about the change log.) The command returns the following information:

`cached_attribute_total_size`

The amount of memory used by the changelog attribute cache, in kilobytes. This number includes additional memory used to manage the cache that is not charged to the individual attribute caches. Consequently, this total is larger than the sum of the memory used by all the individual attribute caches.

`cached_attribute_configured_size`

The maximum amount of memory, in kilobytes, assigned to the changelog attribute cache.

`cached_attribute_click`

The number of times the attribute has been used in a filter that could be processed by the changelog attribute cache. The value is reported as follows:

```
    cached_attribute_click=attrname:#####
cached_attribute_size
```

The amount of memory used for this attribute in the changelog attribute cache. This value is reported in kilobytes as follows:

```
    cached_attribute_size=attrname:######
cached_attribute_candidate_click
```

A list of up to ten most frequently used noncached attributes that have been used in a filter that could have been processed by the changelog attribute cache if all of the attributes used in the filter had been cached. The value is reported as follows:

```
    cached_attribute_candidate_click=attrname:#####
```

You can use this list to help you decide which attributes you want to cache. Typically, you want to put a limited number of attributes into the attribute cache because of memory constraints.

16.16 Troubleshooting error files

When a problem occurs that appears to be related to the IBM Directory Server, you should first check the following files for error messages.

For IBM Directory Server version 4.1 and older: The default location of these files is /var/ldap for Solaris and /tmp for AIX.

```
slapd.errors
cli.error
```

You can change the location of the slapd.errors file (but not the cli.error file) by updating the ibm-slapdErrorLog parameter in the slapd32.conf configuration file.

For IBM Directory Server version 5.1 or later: The default location is /var/ldap for both Solaris and AIX.

```
ibmslapd.log
db2cli.log
```

You can change the location of both of these files by modifying the ibm-slapdErrorLog and ibm-slapdCLIErrors parameters in the ibmslapd.conf.

ibmslapd trace

An ibmslapd trace provides a list of the SQL commands issued to the DB2 database. These commands can help you identify operations that are taking a long time to complete. This information can in turn lead you to missing indexes, or unusual directory topology. To turn the ibmslapd trace on, run the following commands:

```
ldtrc on
ibmslapd –h 4096
```

After you have turned the trace on, run the commands that you think might be giving you trouble. Running a trace on several operations can slow performance, so remember to turn the trace off when you are finished using it:

```
ldtrc off
```

Changing the diagnostic level for error message log files

DB2 error log

On AIX systems or Solaris Operating Environments, the db2diag.log file is located, by default, in the /INSTHOME/sqllib/db2dump directory, where INSTHOME is the home directory of the instance owner.

On Windows NT and Windows 2000 systems, the db2diag.log file is located, by default, in the *x*:\sqllib*instance* directory, where:

► *x:* is the drive where DB2 Data Links Manager is installed.

► *instance* is the name of the instance for which you want to change the diagnostic setting. The instance name in which Data Links Manager is running is DLFM.

The location of the db2diag.log file is controlled by the DB2 server configuration parameter DIAGPATH, so the directory paths on your system might be different from the default paths.

Procedure

You control the level of detailed information that is written to the db2diag.log file by using the DIAGLEVEL configuration parameter and the DLFM_LOG_LEVEL registry value.

DIAGLEVEL

Determines the severity of DB2 diagnostic information recorded in the db2diag.log error log file. Valid values are from 1–4. 1 denotes that a minimal amount of information is to be recorded, and 4 denotes that the maximum amount of information is to be recorded. The default setting is 3. You can increase the amount of error information recorded using the following command: db2 update dbm cfg using DIAGLEVEL 4. This setting should be changed only at the request of IBM service or development for debugging purposes.

DLFM_LOG_LEVEL

Determines the severity of DLFM diagnostic information recorded in the db2diag.log error log file. Its default setting is LOG_ERR. You can increase the amount of error information recorded using the following command:

```
db2set DLFM_LOG_LEVEL=LOG_DEBUG
```

Attention: Increasing the amount of diagnostic output can result in both performance degradation and insufficient storage conditions in your database instance file system. This procedure should only be used when troubleshooting problems requiring the additional diagnostics.

17

Monitoring IBM Tivoli Directory Server

This chapter will cover the monitoring of the IBM Tivoli Directory server covering various monitoring tools and operating system commands that can be used for monitoring ITDS.

17.1 Overview

Like any application, the ability for an administrator to understand what the current state of the application at any given time is critical.

The monitoring of the directory is important from the following perspectives:

1. Security issues: To track unauthorized access and take corrective measures. Let us take some example where we would like to prevent unauthorized access to the directory. There are a lot of instances, whereby the directory server may come under a Denial of Service (DOS) attack. In a DOS attack, anonymous clients send a flood of requests for the server to serve, by opening a set of connections with the server. The clients then disappear and are not available to hear back from the server. The server does not understand what is to be done with the open connection and it waits for the client to return. Thus we have some system resources allocated and kept reserved by the server for a client which is never to return. If there is a pool of such connections a lot of system resources would be wasted and a stage would come, whereby the open connections eat up all the available resources and the server is not able to serve any further requests. We are sure all of us would like to avoid such attacks on our servers. Monitoring the server would help overcoming issues like these and consequently secure our server.

2. Performance issues: To find out reasons of slow or poor performance. Performance of the directory server may be low owing to a lot many reasons:

   – The DB2 buffer pools might not be tuned as per the directory requirements, though there are resources available.

   – The Directory Server caches might not be tuned to the optimum.

   – There might be some important indexes missing.

   – There might be a database reorganization required and many more.

   Out of the above, some problems might be caught or prevented using the monitoring tools. Monitoring tools help us in deriving the optimal values for a set of directory parameters after statistical analysis of the existing workload and resources.

3. Throughput measurement: To derive statistics as like how many searches have completed in a given timeframe, how many additions have completed, how many binds have occurred to the server, how many operations have completed. Keeping track of such figures helps us to calculate the throughput of the server. This throughput might be quite influential for the dependant products that are going to use the directory server.

17.2 Monitoring tools

The ITDS is provided with a set of tools which can be used to monitor the directory server against any anomalies. The ITDS monitoring can be accomplished in many ways as listed below:

► Searching against the base cn=monitor.

► Searching through the changelog database.

► Analyzing log files.

All the monitoring commands have both a GUI interface and the corresponding command line equivalent.

The GUI interface is provided by Web Administration Tool, supplied with the IBM Tivoli Directory Server.

The command-line interface is provided by the following client utilities:

► ldapsearch

► ldapexop

► ibmdirctl

The changelog database is a separate LDAP database, that stores the changes pertaining to the DIT, in LDIF format.

All the relevant log files can be read using the command-line tools or through the GUI, whichever suits one.

> **Note:** Turning on any sort of logging or using a database to log the changes, would hamper the directory performance. The obvious reason is that such activities make the directory do more things than it liked to. For example, the directory server may need to write to 4 places rather than 1. Hence, it is strictly advised that such options be turned on only in the event that the directory server is doing badly and it needs to be tuned. Disable these options, once you have done with you're the problem analysis.

17.2.1 Viewing server state

The first and foremost thing, prior to monitoring ibmslapd, would be to check whether it is still alive, or whether it is in a state where we can run any monitoring tools on it. The server state refers to whether the server is currently running in normal mode, safe mode or it is currently stopped. Let us see how this is accomplished.

Using Web administration tool

To use the Web administration tool:

1. Connect to the server, whose status is to be checked, through the Web administration tool.

2. Select **Server administration** and then click **View server status** in the left hand panel. The General tab is selected by default. If not, select it.

3. The current state of the server is shown in the right panel. We need to watch out for the Server status label.

> **Note:** The current sate of the server can be determined by looking at the icon next to the directory server name or IP in the top of right panel.
>
> ► If the green button is enabled, the server is running.
>
> ► If the yellow button is enabled, the server is running in safe mode.
>
> ► If the red button is enabled, the sever is stopped.

Figure 17-1 shows the relevant portion of the panel.
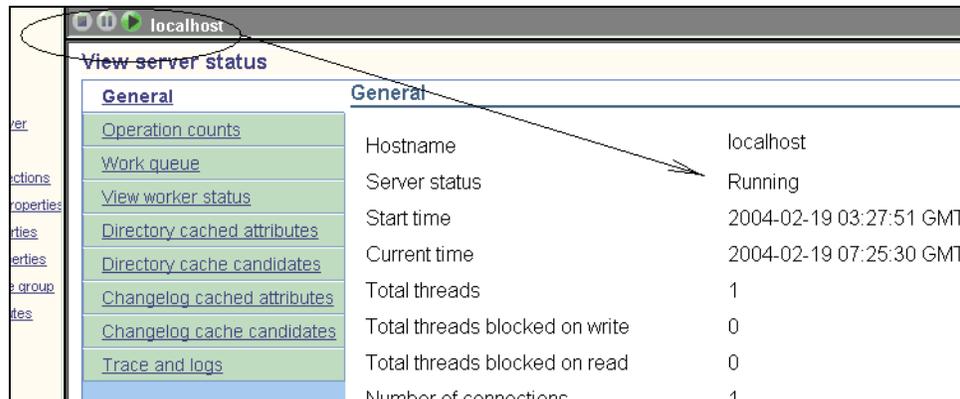
Figure 17-1   Viewing the server status via Web administration tool

Using command line

The following command returns the state of the server:

```
ibmdirctl-D <adminDN> -w <adminPW> status
```

The above command does not say whether the server is running in safe mode or no. We need to confirm it by running a rootDSE search after this command and check for the attribute `ibm-slapdisconfigurationmode` which should be false for a normal mode.

On UNIX:

```
ldapsearch -D <adminDN> -w <adminPW> -s base objectclass=* | grep config
```

On Windows:

```
ldapsearch -D <adminDN> -w <adminPW> -s base objectclass=*
```

Search for the value against the ibm-slapdisconfigurationmode attribute.

Here is an example of how ibmdirctl would be used to get to know the server status and then the status actually verified using the root dse:

```
C:\>ibmdirctl -D <adminDN> -w <adminPW> status
ibmslapd process is not running.
C:\>ibmdirctl -D <adminDN> -w <adminPW> start
Start operation succeeded
C:\>ibmdirctl -D <adminDN> -w <adminPW> status
ibmslapd process is starting.
C:\>ldapsearch -s base objectclass=* | grep configuration
ibm-slapdisconfigurationmode=FALSE
```

If you are just interested in knowing if the server is up (irrespective of whether it is up in normal mode or in configuration mode), you can just check out if the ibmslapd process is currently running using the ps command (`ps -eaf | grep ibmslapd | grep -v grep`), on UNIX of course. In case of Windows, you can see if the service IBM Tivoli Directory Server V5.2 is up and running.

17.2.2 Viewing status of worker threads

This option is required when the server is not performing as expected or performing poorly. This options displays the information on the worker threads that are currently active. The state of a worker thread includes many details like thread number, information about the client it is serving, the type of work request received etc. Performing this activity suspends all the server activity until it is completed. A warning to that effect is displayed, which explains that the time to complete this operation depends on the number of connection and worker threads. Ensure that auditing is enabled before using the below tools for viewing the states of worker threads.

Using Web administration tool

To use this:

1. Connect to the relevant directory server, whose status is to be checked, via the Web administration tool.

2. Select **Server administration** and then click **View server status** server in the left hand panel. Select **View worker status** from the left-hand panel.

3. A warning message appears as shown in Figure 17-2.
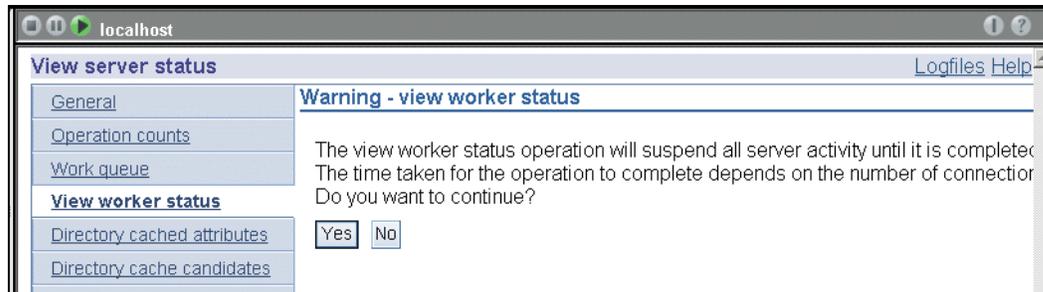
Figure 17-2   Warning while observing the status of the worker threads

4. Click **Yes** to proceed.

In response to the above confirmation, we would see a screen showing us the current status of the worker threads, as seen in Figure 17-3.
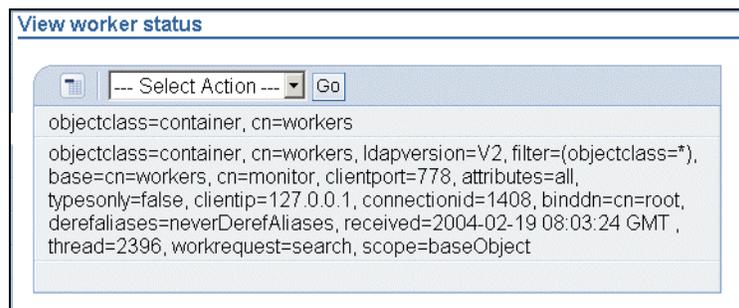
Figure 17-3   The current status of the worker threads

Using command line
In order to retrieve all information related to worker threads that are currently active, issue the following command:

```
ldapsearch -D <adminDN> -w <adminPW> -s base -b cn=workers,cn=monitor
objectclass=*
```

Here is the output that can be expected:

```
cn=workers,cn=monitor
cn=workers
objectclass=container

cn=thread2428,cn=workers,cn=monitor
thread=2428
```

```
ldapversion=V2
binddn=cn=root
clientip=127.0.0.1
clientport=2058
connectionid=1412
received=2004-02-19 08:07:41 GMT
workrequest=search
base=cn=workers,cn=monitor
scope=baseObject
derefaliases=neverDerefAliases
typesonly=false
filter=(objectclass=*)
attributes=all
```

This information is the same as the information displayed on the GUI. Here is what the above attributes mean:

► thread: The number of the worker thread. For example 2428.

► ldapversion: The LDAP version level, either V1 or V2.

► binddn: The DN used to bind to the server.

► clientip: The IP address of the client.

► clientport: The port used by the client.

► connectionid: The number identifying the connection.

► received: The date and time that the work request was received.

► workrequest: The type of work request received and additional information about the request. For example, if the request was a *search*, the following information is also provided:

```
base=cn=workers,cn=monitor
scope=baseObject
derefaliases=neverDerefAliases
typesonly=false
filter=(objectclass=*)
attributes=all
```

That is to say that worker thread 2428 had the responsibility of serving the search request for the base cn=workers, cn=monitor, through the search, which was fired to collect the above information.

17.2.3 Viewing connections information

The connections information is handy in case of problems where the server rejects client connections, for example, if many clients are trying to connect to the server and the number of connections requested is exceeding that permitted by the operating system. The information about the server connections consists

of the connection id, the client ip address which requested the connection, bind dn etc. There are as expected, two ways of viewing this information.

Using Web administration tool

Expand the **Server administration** category in the navigation area as in the previous steps. Click **Manage server connections**. A table containing the following information for each connection is displayed:

► DN: Specifies the DNs of a client connection to the server.

► IP address: Specifies the IP address of the client that has a connection to the server.

► Start time: Specifies the date and time when the connection was made.

► Status: Specifies whether the connection is active or idle. A connection is considered active if it has any operations in progress.

► Ops initiated: Specifies the number of operations requested since the connection was established.

► Ops completed: Specifies the number of operations that have been completed for each connection.

► Type: Specifies whether the connection is secured by SSL or TLS. Otherwise the field is blank.

Figure 17-4 shows the relevant screenshot.

Figure 17-4   Portion of the panel showing the server's connections

> **Note**: The table shown in Figure 17-4 displays up to 20 connections at a given instant of time.

We can specify to have this table displayed by either DN or IP address by expanding the drop-down menu at the top of the panel and making a selection. The default selection is by DN. Similarly we can also specify whether to display the table in ascending or descending order.

Click **Refresh** to update the current connection information.

If you are logged in as the administrator or as a member of the administration group, you have additional selections to disconnect server connections available on the panel. This ability to disconnect server connections enables us to stop denial of service attacks and to control server access. You can disconnect a connection by expanding the drop-down menus and selecting a DN, an IP address or both and clicking Disconnect. Depending on our selections the actions shown in Table 17-1 will occur.

Table 17-1   Disconnection rules.

DN chosen	IP address chosen	Action
<DN Value>	None	All connections bound with the specified DN are disconnected.
None	<IPvalue>	All connections over the specified IP address are disconnected.
<DN Value>	<IPvalue>	All connections bound as the specified DN and over the specified IP address are disconnected.
None	None	This is not a valid condition. You must specify either a DN or an IP or both.

The default value for each of the drop-down menus is None.

To disconnect all server connections except for the one making this request click **Disconnect all**. A confirmation warning is displayed. Click **OK** to proceed with the disconnect action or click **Cancel** to end the action and return to the Manage server connections panel.

Using command line

We can run a search with the searchbase "cn=connections,cn=monitor" to get information about server connections:

```
ldapsearch -D <adminDN> -w <adminPW> -s base -b cn=connections,cn=monitor
objectclass=*
```

This command returns information in the following format:

```
cn=connections,cn=monitor
```

```
connection=3 : 127.0.0.1 : 2004-02-22 06:08:10 GMT  : 1 : 1 : CN=ROOT :  :
```

Note: If appropriate, an SSL or a TLS indicator is added on each connection.

The meaning of the values delimited by ":" in the above output can be very well got by comparing it with the relevant screenshot of the Webadmin (Figure 17-4 on page 554).

To end server connections, issue one of the following commands:

```
# To disconnect a specific DN:
ldapexop -D<adminDN> -w <adminPW> -op unbind -dn cn=john
# To disconnect a specific IP address:
ldapexop -D <adminDN> -w <adminPW> -op unbind -ip 9.182.173.43
#To disconnect a specific DN over a specific IP address:
ldapexop -D <adminDN> -w <adminPW> -op unbind -dn cn=john -ip 9.182.173.43
#To disconnect all connections:
ldapexop -D <adminDN> -w <adminPW> -op unbind -all
```

Does not this option give the great advantage of killing the connections, which are seen harmful for our directory performance/stability. But as the chapter title suggests, it would need a constant monitoring to find out and disconnection unwanted, harmful connection. Do not take this statement to be contradictory to the note about performance click, when the monitoring is on. The performance click is felt only when the logging and/or the changelog database come into the picture and does not relate to the searches. The searches will not slow down a server.

17.2.4 Viewing other general information about the directory server

Except the connection and worker thread information, a lot of other general information about the server can be fetched. There is voluminous amount of attributes that can be fetched from the server. As usual we have two ways of fetching these attributes.

Using Web administration tool
To use this:

1. Connect to the required directory server, using the Web Administration tool.

2. Click **View server status**. This panel has nine tabs. These are:

    – General tab: This tab provides the generic information pertaining to the server, as like:

      • Hostname: The host name of the LDAP server.

- Server status: The server status as to whether it is currently Running, Stopped, or Running in configuration only mode. We can determine the server status at any time by viewing the three icons displayed in the upper left corner of the server status area.

- Start time: The time the server was started. The start time is in the format: year-month-day hour:minutes:seconds GMT.

- Current time: The time at the instant the General tab was clicked or the time when the Refresh button has been click. The current time is in the format: year-month-day hour:minutes:seconds GMT.

- Total threads: The number of worker threads being used by the server.

- Total threads blocked on write: The number of threads sending data back to the client.

- Total threads blocked on read: The number of threads reading data from the client.

- Number of connections: The number of connections, currently active.

- Total connections: The total number of connections since the server was started.

- Number of entries sent: The number of entries sent by the server since the server was started.

- Percentage of entry cache used: The percentage of entry cache currently used. This value is not displayed in configuration only mode.

- Percentage of search filter cache used: The percentage of search filter cache currently used. This value is not displayed in configuration only mode.

- ACL cache: A Boolean value indicating that the ACL cache is active (TRUE) or inactive (FALSE). This value is not displayed in configuration only mode.

- Maximum ACL cache size: The maximum number of entries allowed in the ACL cache. This value is not displayed in configuration only mode.

- Bypass alias dereferencing: The server runtime value that indicates if alias processing can be bypassed. It displays true, if no alias object

- Total number of SSL connections: The total number of SSL connections since the server was started.

- Total number of TLS connections: The total number of TLS connections since the server was started.

– Operations counts: This tab counts the different type of operations requested/competed with the server:

- Number of operations requested: The number of requests initiated since the server was started.

- Number of operations completed: The number of requests completed, since the server was started.

- Number of search operations requested: The number of searches initiated since the server was started.

- Number of search operations completed: The number of searches completed, since the server was started.

- Number of bind operations requested: The number of bind requests since the server was started.

- Number of bind operations completed: The number of bind requests completed since the server was started.

- Number of unbind operations requested: The number of unbind requests since the server was started.

- Number of unbind operations completed: The number of unbind requests completed since the server was started.

- Number of add operations requested: The number of add requests since the server was started.

- Number of add operations completed: The number of add requests completed since the server was started.

- Number of delete operations requested: The number of delete requests since the server was started.

- Number of delete operations completed: The number of delete requests completed since the server was started.

- Number of modify RDN operations requested: The number of modify RDN requests since the server was started.

- Number of modify RDN operations completed: The number of modify RDN requests completed since the server was started.

- Number of modify operations requested: The number of modify requests since the server was started.

- Number of modify operations completed: The number of modify requests completed since the server was started.

- Number of compare operations requested: The number of compare requests since the server was started.

- Number of compare operations completed: The number of compare requests completed since the server was started.
- Number of abandon operations requested: The number of abandon requests since the server was started.
- Number of abandon operations completed: The number of abandon requests completed since the server was started.
- Number of extended operations requested: The number of extended requests since the server was started.
- Number of extended operations completed: The number of extended requests completed since the server was started.
- Number of unknown operations requested: The number of unknown requests since the server was started.
- Number of unknown operations completed: The number of unknown requests completed since the server was started.

– Work queue: This tab gives the current status on the work queue. Do not confuse this with the status of the worker threads.
  - Number of worker threads available: The number of worker threads available for work.
  - Depth of the work queue: The current size of the work queue.
  - Largest size of the work queue: The largest size that the work queue has ever reached.
  - Number of connections closed by automatic connection cleaner: The number of idle connections closed by the automatic connection cleaner.
  - Number of times the automatic connection cleaner has run: The number of times the automatic connection cleaner has run.
  - Emergency thread currently active: The indicator of whether the emergency thread is running.
  - Number of times the emergency thread has been activated: The number of times the emergency thread has been activated.
  - Last time the emergency thread was activated: The last time the emergency thread was activated.

– Directory cached attributes: This tab provides the information pertaining to the directory's cached attributes.
  - Attribute: The name of the attribute.
  - Number of cache clicks: The number of times the attribute filter has been used after it was cached.

- Cache size: The amount of memory used by the attribute.

- Cached attribute total size (in kilobytes): The amount of memory being used by the cache. This number includes additional memory used to manage the cache, that is not charged against the individual attributes. Consequently, this total is larger than the total of the individual attribute memory usage.

- Cached attribute configured size: The maximum amount of memory in bytes assigned to this cache.

– Directory cache candidates: This tab gives information on which attributes are good candidates for being put in the attribute cache.

- Attribute: The name of the attribute.

- Number of clicks: The number of times the attribute filter has been used.

– Changelog cached attributes: This tab gives information on the cached attributes pertaining to the changelog.

- Attribute: The name of the attribute.

- Number of cache clicks: The number of times the attribute filter has been used after it was cached.

- Cache size: The amount of memory used by the attribute.

- Cached attribute total size (in kilobytes): The amount of memory being used by the cache. This number includes additional memory used to manage the cache, that is not charged against the individual attributes. Consequently this total is larger than the total of the individual attribute memory usage.

- Cached attribute configured size: The amount of memory assigned to this cache.

– Changelog cache candidates: This tab provides information on the attributes that are good candidates for being kept in the changelog cache.

- Attribute: The name of the attribute.

- Number of clicks: The number of times the attribute filter has been used.

– Trace and logs: This tab provides information pertaining to the server trace and the relevant logs.

- Trace enabled: The current trace value for the server. TRUE, if collecting trace data, FALSE, if not collecting trace data.

- Trace message level: The current ldap_debug value for the server. The value is in hexadecimal form, for example, 0x0=0, 0xffff=65535.

- Trace message log: The name of the file that contains the trace output. If the value is stderr, the output is displayed in the command window where the LDAP server was started. If the server was not started from the command line, no data is displayed.

- Number of messages added to server logs: The number of error messages recorded since the server started.

- Number of messages added to CLI error log: The number of DB2 error messages recorded since the server started.

- Number of messages added to audit log: The number of messages recorded by the audit log since the server started.

- Number of error messages added to audit log: The number of failed operation messages recorded by the audit log.

Using command line

The above information about the server can be obtained using a ldapsearch against base cn=monitor. The command for doing a monitor search is:

```
ldapsearch -D <adminDN> -w <adminPW> -s base -b cn=monitor objectclass=*
```

The information returned by the above search is as follows:

► version=IBM Tivoli Directory (SSL), Version 5.2.

► totalconnections: The total number of connections since the server was started.

► total_ssl_connections: The total number of SSL connections since the server was started.

► total_tls_connections: The total number of TLS connections since the server was started.

► currentconnections: The number of active connections.

► maxconnections: The maximum number of active connections allowed.

► writewaiters: The number of threads sending data back to the client.

► readwaiters: The number of threads reading data from the client.

► opsinitiated: The number of requests since the server was started.

► livethreads: The number of worker threads being used by the server.

► opscompleted: The number of completed requests since the server was started.

► entriessent: The number of entries sent by the server since the server was started.

- ► searchesrequested: The number of searches requested since the server was started.

- ► searchescompleted: The number of searches completed since the server was started.

- ► bindsrequested: The number of bind operations requested since the server was started.

- ► bindscompleted: The number of bind operations completed since the server was started.

- ► unbindsrequested: The number of unbind operations requested since the server was started.

- ► unbindscompleted: The number of unbind operations completed since the server was started.

- ► addsrequested: The number of add operations requested since the server was started.

- ► addscompleted: The number of add operations completed since the server was started.

- ► deletesrequested: The number of delete operations requested since the server was started.

- ► deletescompleted: The number of delete operations completed since the server was started.

- ► modrdnsrequested: The number of modify RDN operations requested since the server was started.

- ► modrdnscompleted: The number of modify RDN operations completed since the server was started.

- ► modifiesrequested: The number of modify operations requested since the server was started.

- ► modifiescompleted: The number of modify operations completed since the server was started.

- ► comparesrequested: The number of compare operations requested since the server was started.

- ► comparescompleted: The number of compare operations completed since the server was started.

- ► abandonsrequested: The number of abandon operations requested since the server was started.

- ► abandonscompleted: The number of abandon operations completed since the server was started.

- extopsrequested: The number of extended operations requested since the server was started.

- extopscompleted: The number of extended operations completed since the server was started.

- unknownopsrequested: The number of unknown operations requested since the server was started.

- unknownopscompleted: The number of unknown operations completed since the server was started.

- slapderrorlog_messages: The number of server error messages recorded since the server was started or since a reset was performed.

- slapdclierrors_messages: The number of DB2 error messages recorded since the server was started or since a reset was performed.

- auditlog_messages: The number of audit messages recorded since the server was started or since a reset was performed.

- auditlog_failedop_messages: The number of failed operation messages recorded since the server was started or since a reset was performed.

- filter_cache_size: The maximum number of filters allowed in the cache.

- filter_cache_current: The number of filters currently in the cache.

- filter_cache_click: The number of filters found in the cache.

- filter_cache_miss: The number of filters not found in the cache.

- filter_cache_bypass_limit: Search filters that return more entries than this limit are not cached.

- entry_cache_size: The maximum number of entries allowed in the cache.

- entry_cache_current: The number of entries currently in the cache.

- entry_cache_click: The number of entries found in the cache.

- entry_cache_miss: The number of entries not found in the cache.

- acl_cache: A Boolean value indicating that the ACL cache is active (TRUE) or inactive (FALSE).

- acl_cache_size: The maximum number of entries in the ACL cache.

- cached_attribute_total_size: The amount of memory used by the directory attribute cache.

- cached_attribute_configured_size: The amount of memory assigned to the directory attribute cache.

- currenttime: The current time on the server. The current time is in the format: year-month-day hour:minutes:seconds GMT.

- ► starttime: The time the server was started. The start time is in the format: year-month-day hour:minutes:seconds GMT

- ► trace_enabled: The current trace value for the server. TRUE, if collecting trace data, FALSE, if not collecting trace data. See ldaptrace for information about enabling and starting the trace function.

- ► trace_message_level: The current ldap_debug value for the server. The value is in hexadecimal form, for example: 0x0=0, 0xffff=65535

- ► trace_message_log: The current LDAP_DEBUG_FILE environment variable setting for the server.

- ► en_currentregs: The current number of client registrations for event notification.

- ► en_notificationssent: The total number of event notifications sent to clients since the server was started.

- ► bypass_deref_aliases: The server runtime value that indicates if alias processing can be bypassed. It displays true, if no alias object exists in the directory, and false, if at least one alias object exists in the directory.

- ► available_workers: The number of worker threads available for work.

- ► current_workqueue_size: The current depth of the work queue.

- ► largest_workqueue_size: The largest size that the work queue has ever reached.

- ► idle_connections_closed: The number of idle connections closed by the Automatic Connection Cleaner.

- ► auto_connection_cleaner_run: The number of times that the Automatic Connection Cleaner has run.

- ► emergency_thread_running: The indicator of whether the emergency thread is running.

- ► totaltimes_emergency_thread_run: The number of times the emergency thread has been activated.

- ► lasttime_emergency_thread_run: The last time the emergency thread was activated.

Now let us see some examples as to how the above attributes help in tuning the directory.

The following sections show examples of using values returned by the ldapsearch command with "cn=monitor" to calculate the throughput of the server and the number of add operations completed on the server in a certain timeframe.

Throughput example: The following example shows how to calculate the throughput of the server by monitoring the server statistic called opscompleted, which is the number of operations completed since the LDAP server started.

Suppose the values for the opscompleted attribute obtained by issuing two ldapsearch commands to monitor the performance statistics, one at time t1 and the other at a later time t2, are opscompleted (t1) and opscompleted(t2) respectively. The average throughput at the server during the interval between t1 and t2 can be calculated as:

```
(opscompleted(t2) - opscompleted(t1) - 3)/(t2 -t1)
```

Three is subtracted to account for the number of operations performed by the ldapsearch command itself.

Workload example: The monitor attributes can be used to characterize the workload, similar to the throughput example, but split out by type of operation. For example, we can calculate the number of add operations that were completed in a certain amount of time.

Suppose the values for the addscompleted attribute obtained by issuing two ldapsearch commands to monitor the performance statistics, one at time t1 and the other at a later time t2, are addscompleted (t1) and addscompleted(t2) respectively. The number of add operations completed on the server during the interval between t1 and t2 can be calculated as:

```
(addscompleted(t2) - addscompleted(t1) - 3)/(t2 -t1)
```

Three is subtracted to account for the number of operations performed by the ldapsearch command itself.

Similar calculations can be done for other operations, such as searchescompleted, bindscompleted, deletescompleted, and modifiescompleted.

If you want to know the cache settings suitable for your environment, you note down the cache settings at this point of time and also the relevant performance/throughput. Then change the cache settings and note down the throughput again.

Once you obtain a set of throughputs in the above manner, prepare the trend/charts whereby you get to know the optimal value for the cache settings. Likewise, we can tune the other parameters too and get their optimal value. Please have a hands on with cn=monitor searches as they are the most powerful means of monitoring the directory for performance.

17.2.5 Analyzing changelog

Prior to analyzing the changelog, lets see why do we need to have a changelog in place. The change log is maintained in the form of a separate database as compared to the LDAP database, where the DIT is stored. It is used to record changes to the schema or directory entries in the typical LDAP entry structure that can be retrieved through the LDAP API. The change log records all update operations that happen at the directory server: add, delete, modify, and modrdn. The change log enables an IBM Tivoli Directory Server client application to retrieve a set of changes that have been made to an IBM Tivoli Directory Server database. The client might then update its own replicated or cached copy of the data.

Viewing the changelog using the Web Administration console

To do this:

1. Log into the directory server using the Web administration tool.

2. Click the tab **Directory management**. Click **Manage entries**. Then expand the suffix cn=changelog.

3. All the recorded changes to the DIT will appear below it in the format changenumber=<integer>.

Figure 17-5 shows the screenshot of the changelog contents, with just one change having been recorded, in this database.
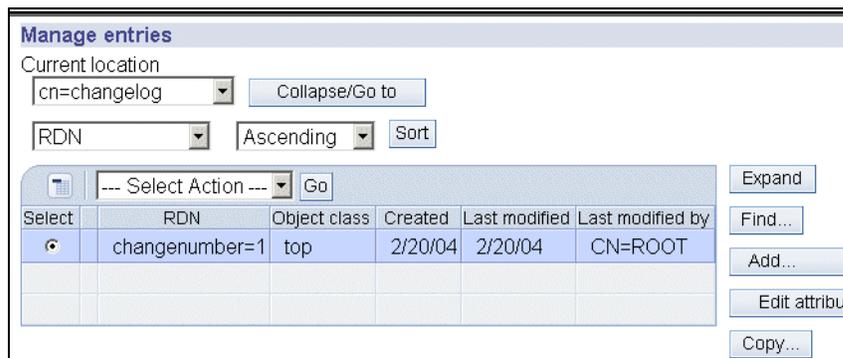
Figure 17-5   Contents of the change log

If you want to see details on a particular change that was performed against the directory server, you may click the **Edit attributes** button, shown above.

Viewing the changelog using ldapsearch

All the changenumbers under the suffix cn=changelog can be retrieved by the following command:

```
ldapsearch -D cn=<adminDN> -w <adminPW> -b cn=changelog changenumber=*
```

Also a particular change number can be requested using:

```
ldapsearch -D cn=<adminDN> -w <adminPW> -b cn=changelog
changenumber=<integer>
```

It returns information in the following format:

```
changenumber=1,cn=changelog
objectclass=top
objectclass=changelogentry
objectclass=ibm-changelog
changenumber=1
targetdn=o=ibm,c=ind
changetype=modify
changetime=20031217094348
ibm-changeInitiatorsName=CN=ROOT
changes=replace: businesscategory
businesscategory: something
```

> **Note**: Enabling the changelog is seen as a performance bottleneck, because the directory server would have to write to the LDAP database as well as log the relevant information in the changelog database. Therefore it is advisable to have the changelog enabled only in the event that a problem is being debugged or if another application in your organization (that is, a meta-directory tool) required it to be on.

17.2.6 Analyzing log files

In this section we analyze the log files.

Audit log

Audit logging is used to improve the security of the directory server. A default audit plug-in is provided with the directory server. Depending upon the audit configuration parameters, this plug-in might log an audit entry in the default or specified audit log for each LDAP operation the server processed. The system administrator can use the activities stored in the audit log to check for suspicious patterns of activity, in an attempt to detect security violations. If security is violated, the audit log can be used to determine how and when the problem occurred and perhaps the amount of damage done. This information is very useful, both for recovery from the violation and, possibly, in the development of better security measures to prevent future problems. We can also write our own

audit plug-ins to either replace, or add more processing to, the default audit plug-in.

By default the audit log is disabled.

> **Note**: Members of the administrative group can view the audit log and the associated settings but not modify them. Only the root administrator is allowed to access, change or clear the audit log files.

The audit log can be configured to track various activities happening against the directory server like attempted logins, requested operations, the timestamp of the operations etc. It is a plain text file created in the /var/ldap directory in case of unix systems (ldapinstalldir\ibm\ldap\var in the case of windows). The audit log file is a crucial tool in monitoring the directory activities.

In order to start using the Audit Log, it first needs to enabled. As expected, there are two ways of enabling the audit log.

Using Web administration tool

To use this:

1. Expand **Logs** in the navigation area, click **Modify audit log settings**.

2. Select **Enable audit logging** to use the audit log utility.

3. Select the **Audit version** you want to use. Version 1 maintains previous audit logging capabilities for any applications that parse the audit log. Version 2 enables you to log extended operations, however, you might need to modify existing applications that parse the audit log.

4. Select to either log Only failed attempts of the selected operations or to log All attempts of the selected operations.

5. Enter the Path and file name for the audit log. The audit log can also be directed to something other than a file, for example, a line printer.

6. Select the operations you wish to log. Consult the field help for additional information about the various operations you can log.

   – Bind - records connections to the server

   – Unbind - records disconnections from the server

   – Search - records LDAP search operations performed by any client

   – Add - records additions to LDAP

   – Modify - records modifications to LDAP

   – Delete - records deletions from LDAP

- – Modify RDN - records modifications made to RDNs
- – Event notification - records event notifications
- – Extended operations- records extended operations performed against the server

> **Note**: If you have selected audit Version 1, selecting Extended operations does not activate this function. You must select audit version 2 for the auditing of extended operations to work.

7. Click **OK** to apply the changes or click **Cancel** to return to the IBM Tivoli Directory Server Web Administration Welcome panel without making any changes.

Figure 17-6 on page 570 shows the screenshot of the relevant panel, where you would be doing these changes.
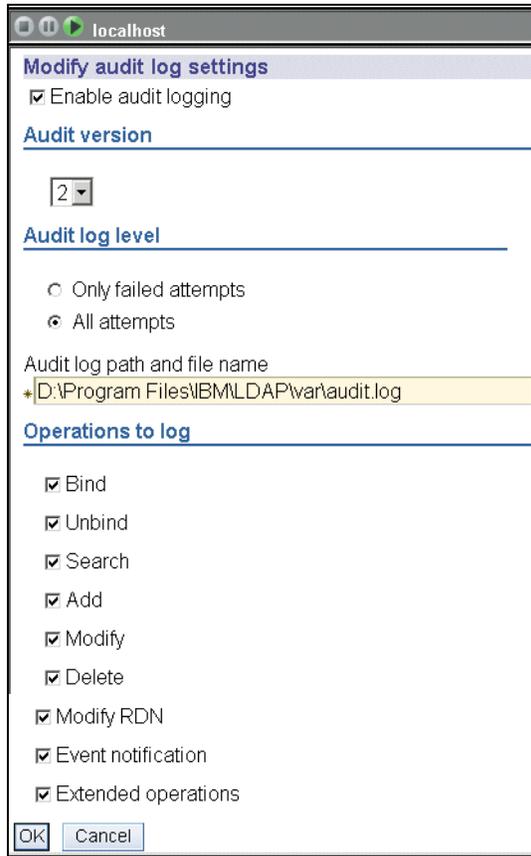
Figure 17-6   Panel to enable/disable the audit log

Using the command line

The similar operations can be done via the command using our very own ldapmodify command. Here is how:

```
ldapmodify -D <adminDN> -w <adminPW> -i <filename>
```

Where <filename> contains:

```
dn: cn=audit, cn=localhost
changetype: modify
replace: ibm-audit
ibm-audit: true
-
replace: ibm-auditadd
ibm-auditadd: {TRUE|FALSE}
#select TRUE to enable, FALSE to disable
```

```
-
replace: ibm-auditbind
ibm-auditbind: {TRUE|FALSE}
#select TRUE to enable, FALSE to disable
-
replace: ibm-auditdelete
ibm-auditdelete: {TRUE|FALSE}
-
replace: ibm-auditextopevent
ibm-auditextopevent: {TRUE|FALSE}
#select TRUE to enable, FALSE to disable
-
replace: ibm-auditfailedoponly
ibm-auditfailedoponly: {TRUE|FALSE}
#select TRUE to enable, FALSE to disable
-
replace: ibm-auditlog
ibm-auditlog: <newpathname>
-
replace: ibm-auditmodify
ibm-auditmodify: {TRUE|FALSE}
#select TRUE to enable, FALSE to disable
-
replace: ibm-auditmodifydn
ibm-auditmodifydn: {TRUE|FALSE}
#select TRUE to enable, FALSE to disable
-
replace: ibm-auditsearch
ibm-auditsearch: {TRUE|FALSE}
#select TRUE to enable, FALSE to disable
-
replace: ibm-auditunbind
ibm-auditunbind: {TRUE|FALSE}
#select TRUE to enable, FALSE to disable
-
replace: ibm-auditversion
ibm-auditversion: {1|2}
#select 2, if you are enabling audit for extended operations
-
replace: ibm-auditExtOp
ibm-auditExtOp: {TRUE|FALSE}
#select TRUE to enable, FALSE to disable
```

Note: If you are using audit logging in Configuration only mode, the DN specified is dn: cn=audit, cn=configuration. Any changes made to this DN are overwritten with the dn: cn=audit, cn=localhost values when the server is started in normal mode.

Disabling the audit log

Again, we are going to see two ways to disable audit logging.

Using Web Administration

To use this:

1. Expand **Logs** in the navigation area, click **Modify audit log settings**.

2. Deselect **Enable audit logging**.

3. Click **OK** to apply the changes or click **Cancel** to return to the IBM Tivoli
   Directory Server Web Administration Welcome panel without making any
   changes.

Using the command line

The similar operations can be done via the command using our very own
ldapmodify command. Here is how:

```
ldapmodify -D <adminDN> -w <adminPW> -i <filename>
```

Where <filename> contains:

```
dn: cn=audit, cn=localhost
changetype: modify
replace: ibm-audit
ibm-audit: false
You do not need to deselect the individual operations to disable auditing.
Just running the above ldapmodify should suffice.
```

> **Note**: If you are using audit logging in Configuration only mode, the DN
> specified is dn: cn=audit, cn=configuration. Any changes made to this DN
> are overwritten with the dn: cn=audit, cn=localhost values when the server
> is started in normal mode.

Viewing the audit log

The audit log displays, log entries, chronologically. Each non-message entry
contains a general information header followed by operation-specific data. For
example:

```
2000-03-23-16:01:01.345-06:00--V3 Bind--bindDN:cn=root
--client:9.1.2.3:12345--
ConnectionID:12--received:2000-03-23-16:01:01.330-06:00 --success
name: cn=root
authenticationChoice: simple
If the audit version is version 2 the header contains __AuditV2--__.
AuditV2--2003-07-22-09:39:54.421-06:00DST--V3 Bind--bindDN:
```

```
cn=root--client: 127.0.0.1:8196--connectionID: 3--received:
2003-07-22-09:39:54.421-06:00DST--Success
```

The header is in the following format:

- ► Timestamp 1 __--__: The local time the entry is logged, that is, the time the request was processed. The timestamp is in the format YYYY-MM-DDHH: MM:SS.mmm=(or-)HH:MM. The =(or=)HH:MM is UTC offset. mmm is milliseconds.

- ► Version number+[SSL]+[unauthenticated or anonymous] Operation __--__: Shows the LDAP request that was received and processed. Version number is either V2 or V3. SSL displays only when SSL was used for the connection. unauthenticated or anonymous displays to indicate whether the request was from an unauthenticated or anonymous client. Neither unauthenticated nor anonymous are logged, in case the request is from an authenticated client.

- ► bindDN: Shows the bind DN. For V3 unauthenticated or anonymous requests, this field is <*CN=NULLDN*>.

- ► client:Client IP address:Port number __--__: Shows the client IP address and port number.

- ► ConnectionID: xxxx __--__: Is used to group all the entries received in the same connection, meaning between the bind and unbind, together.

- ► received: Timestamp 2 __--__: Is the local time when the request was received, or to be more specific, the beginning time when the request was processed. Its format is the same as Timestamp 1.

Result or Status string Shows the result or status of the LDAP operation. For the result string, the textual form of the LDAP resultCode is logged, for example, success or operationsError, instead of 0 or 1.

Operation-specific data follows the header and displays operation-specific data, for example,

```
Bind operations
    name: Y249bWFuYWdlcgOK
    authenticationChoice: simple
Add operations
    entry: cn=Jim Brown, ou=sales,o=ibm_us,c=us
    attributes: objectclass, cn, sn, telphonenumber
Delete operations
    entry: cn=Jim Brown, ou=sales,o=ibm_us,c=us
Modify operations
    object: cn=Jim Brown, ou=sales,o=ibm_us,c=us
    add: mail
    delete: telephonenumber
```

Now let us see how we can see the contents of the audit log.

Using Web Administration

To do this:

1. Expand Logs in the navigation area, click **View audit log**.

2. The panel displays the first page of the audit log.The navigation arrows at the bottom of the panel enable you to go to the Next page or to the Previous page. From the menu, you can select a specific page, for example Page 6 of 16, and click **Go** to display that page of the audit log. You can:

   a. Click **Refresh** to update the entries in the log.

   b. Click **Clear log** to delete all entries in the audit log.

   c. Click **Close** to return to the IBM Tivoli Directory Server Web Administration Welcome panel.

Figure 17-7 shows the relevant screenshot of the panel you will see, when you view the audit log via the Web Administration tool.
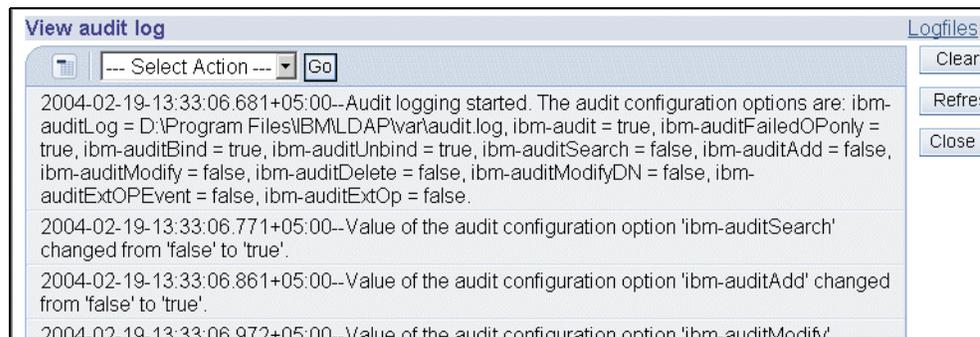
Figure 17-7   Contents of the audit log

Using the command line

To view the audit log through the command line, issue the following command (for UNIX):

```
more /var/ldap/audit.log
```

Where /var/ldap/audit.log is the default path for the audit log.

> **Note**: /var/ldap/audit.log is the default audit log for UNIX systems and ldapinstalldir\var\audit.log is the default audit log for Windows systems. The above command will not work if you have set a Custom path for the audit log.

To view and clear the audit log dynamically:

```
ldapexop -D <adminDN> -w <adminPW> -op readlog -log audit -lines all
```

```
                    ldapexop -D <adminDN> -w <adminPW> -op clearlog -log audit
```

The ldapexop tool can be used to fetch the whole or some required number of lines from the audit log file. The command for the same is as shown below:

```
                    ldapexop -D <adminDN> -w <adminPW> -op readlog -log audit -lines all
```

It returns the audit log as follows:

```
authenticationChoice: simple
AuditV2--2003-12-17-14:53:36.554-05:00--V3 Unbind--bindDN: cn=root--client:
127.0.0.1:19728--connectionID: 33--received:
2003-12-17-14:53:36.554-05:00--Success
controlType: 2.16.840.1.113730.3.4.2
criticality: false
AuditV2--2003-12-17-14:53:36.654-05:00--V3 Unbind--bindDN: cn=root--client:
127.0.0.1:18704--connectionID: 31--received:
2003-12-17-14:53:36.654-05:00--Success
controlType: 2.16.840.1.113730.3.4.2
criticality: false
AuditV2--2003-12-17-14:54:33.065-05:00--V3 Bind--bindDN: cn=root--client:
9.24.104.185:20240--connectionID: 34--received:
2003-12-17-14:54:33.065-05:00--Success
name: cn=root
authenticationChoice: simple
```

Here is what is expected, when we clear the log and then try to read its contents:

```
C:\>ldapexop -D <adminDN> -w <adminPW> -op clearlog -log audit
audit log file cleared.
C:\>ldapexop -D -D <adminDN> -w <adminPW> -op readlog -log audit -lines all
   Feb 22 00:23:44 2004 Log file cleared.
   AuditV2--2004-02-22-00:23:44.645+05:00--V3 extended operation--bindDN:
   cn=root--client: 127.0.0.1:3588--connectionID: 3--received:
   2004-02-22-00:23:44.645+05:00--SuccessOID: 1.3.18.0.2.12.20
   AuditV2--2004-02-22-00:23:44.655+05:00--V3 Unbind--bindDN: cn=root--
   client: 127.0.0.1:3588--connectionID: 3--received:
   2004-02-22-00:23:44.655+05:00--Success
   AuditV2--2004-02-22-00:23:52.416+05:00--V3 Bind--bindDN: cn=root--
   client: 127.0.0.1:3844--connectionID: 4--received:
   2004-02-22-00:23:52.416+05:00--Success
   name: cn=root
   authenticationChoice: simple
   AuditV2--2004-02-22-00:23:52.416+05:00--V3 extended operation--bindDN:
   cn=root--client: 127.0.0.1:3844--connectionID: 4--received:
   2004-02-22-00:23:52.416+05:00--Success
```

ibmslapd Error log

The errors pertaining to the server operations are logged in what is known as the ibmslapd error log. This file can also be handy in some cases. For example, if

you try to add entries with some object class violations, they can very easily be noticed by means of this error log. You can note error messages like this in the slapd error log:

```
Feb 15 04:26:31 2004   The required attribute sn is missing for entry
cn=user1,o=ibm,c=us.
Feb 15 04:26:31 2004   Entry cn=user1,o=ibm,c=us violates the schema
definition.
```

There are other errors too like the Master unable to contact the Replica for a given reason. These also appear in the ibmslapd error log. So if there is any problem pertaining to the server that you need to look at, feel free to go through the ibmslapd error log and there might be a hint of the problem out there.

> **Note:** The error log, ibmslapd.log, is enabled by default.

To modify error log settings, there are again two ways to do it. First, We will see the necessary changes through the Web Administration Tool:

1. Expand Server administration in the navigation area, click **Logs**, click **Modify error log settings**.

2. Enter the path and file name for the error log. Ensure that the path is valid. If the file does not exist, it is created. The error log can also be directed to something other than a file, for example, a line printer.

3. Select either **Low**, **Medium**, or **High** for the level of error logging.

> **Note:** If you specify a file that is not an acceptable file name (for example, invalid syntax or if the server does not have the rights to create and/or modify the file), the attempt fails with the following error: LDAP Server is unwilling to perform the operation.

   a. Low logs the least amount of error information, for example:

   ```
   Mar 29 11:03:23 2002 IBM Directory, Version 5.2
   slapd started.
   ```

   b. Medium logs a medium amount of error information, for example:

   ```
   Mar 29 11:07:51 2002 Configuration read securePort 636.
   Mar 29 11:07:51 2002 Plugin of type PREOPERATION is successfully loaded
   from libDSP.dll.
   Mar 29 11:07:51 2002 Plugin of type DATABASE is successfully loaded from
   C:\Program Files\IBM\LDAP/bin/libback-rdbm.dll.
   Mar 29 11:08:11 2002 Non-SSL port initialized to 389.
   Mar 29 11:08:12 2002 IBM Directory, Version 5.2
   slapd started.
   ```

c. High logs the most amount of error information, for example:

```
Mar 29 11:04:05 2002 Configuration read securePort 636.
Mar 29 11:04:05 2002 Configuration read cipher specifications mask to be
12288.
Mar 29 11:04:05 2002 Plugin of type PREOPERATION is successfully loaded
from libDSP.dll.
Mar 29 11:04:05 2002 Plugin of type DATABASE is successfully loaded from
C:\Program Files\IBM\LDAP/bin/libback-rdbm.dll
Mar 29 11:04:24 2002 Configuration file successfully read.
Mar 29 11:04:24 2002 Non-SSL port initialized to 389.
Mar 29 11:04:25 2002 IBM Directory, Version 5.2
slapd started.
```

4. Click **OK** to apply the changes or click **Cancel** to return to the IBM Tivoli Directory Server Web Administration Welcome panel without making any changes.

5. Click **OK** to return to the IBM Tivoli Directory Server Web Administration Welcome panel.

Figure 17-8 shows the relevant screenshot.
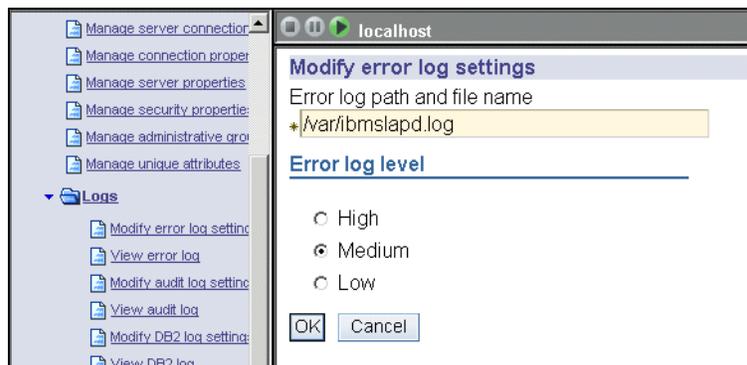
Figure 17-8   ibmslapd error log settings

Using the command line

Issue the command:

```
ldapmodify -D <adminDN <-w >adminPW> -i <filename>
where <filename> contains:
      dn: cn=Configuration
      changetype: modify
      replace: ibm-slapdErrorLog
      ibm-slapdErrorLog: <newpathname>
      -
      replace: ibm-slapdSysLogLevel
```

```
ibm-slapdSysLogLevel: {l | m | h}
```

To update the settings dynamically, issue the following **ldapexop** command:

```
ldapexop -D <adminDN> -w <adminPW> -op readconfig -scope entire
```

The **ldapexop** command updates only those attributes that are dynamic. For more information on which attributes can be updated dynamically and which not, you can go through Chapter 10, "Client tools" on page 237.

Viewing the error log

Use the following procedures to view the error log.

Using Web Administration

To use this:

1. Expand **Logs** in the navigation area, then click **View error log**.

2. The panel displays the first page of the error log and the navigation arrows at the bottom of the panel enable you to go to the Next page or to the Previous page. From the menu, you can select a specific page, for example Page 6 of 16, and click **Go** to display that page of the error log. You can:

   a. Click **Refresh** to update the entries in the log.

   b. Click **Clear log** to delete all entries in the administration daemon error log.

   c. Click **Close** to return to the IBM Tivoli Directory Server Web Administration Welcome panel.

Figure 17-9 shows the relevant screenshot, which shows a portion of the messages logged.
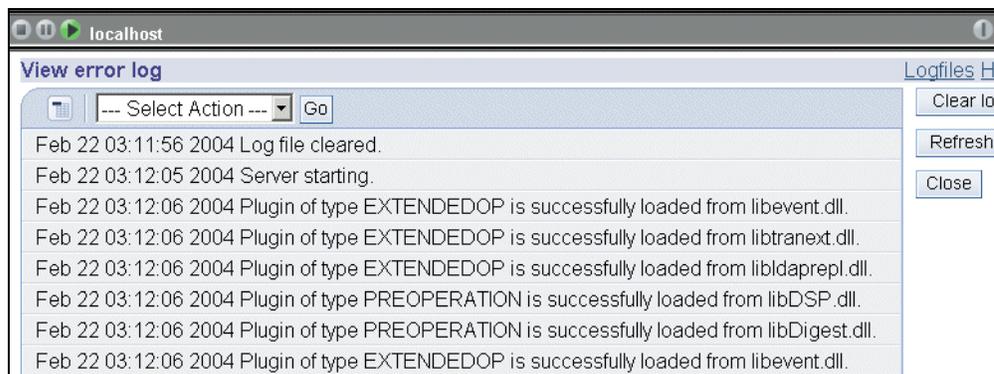
Figure 17-9   Contents of the ibmslapd error log file

Using the command line

To view the error log, issue the following command (on UNIX):

```
more /var/ldap/ibmslapd.log
```

Where /var/ldap/ibmslapd.log is the default path for the ibmslapd error log.

> **Note:** /var/ldap/ibmslapd.log is the default error log for UNIX systems and
> ldapinstalldir\var\ibmslapd.log is the default error log for Windows systems.

To view and clear the error log dynamically:

```
ldapexop -D <adminDN> -w <adminPW> -op readlog -log slapd -lines all
ldapexop -D <adminDN> -w <adminPW> -op clearlog -log slapd
```

DB2 error log

In addition to the ibmslapd.log file, which can be accessed through the Web
Administration Tool, DB2 errors are logged in the db2cli.log file. Both files are
located in the var subdirectory of the IBM Tivoli Directory Server installation
directory on Windows platforms. There exist a lot of parameters at the DB2 level,
which will enhance our directory server's performance. In case any of the
parameter is set below/above the acceptable limits, the relevant message will be
logged into the db2cli.log.

> **Note:** The var subdirectory might include other DB2 files.

Server errors, by default, are logged in the \var\ibmslapd.log file.

DB2 errors are, by default, logged in the \var\db2cli.log file.

Modifying DB2 error log settings

As expected, there are two ways of modifying the settings of the DB2 Error log:

1. Expand **Logs** in the navigation area, click **Modify DB2 log settings**.
2. Enter the path and file name for the error log. Typically this is the db2cli.log
   file located in the /var/ldap directory. Ensure that the path is valid. If the file
   does not exist, it is created.

> **Note:** /var/ldap/db2cli.log is the default DB2 error log for UNIX systems
> and ldapinstalldir\var\db2cli.log is the default DB2 error log for Windows
> systems.

3. Click **OK** to apply the changes or click **Cancel** to return to the IBM Tivoli Directory Server Web Administration Welcome panel without making any changes.

4. Click **OK** to return to the IBM Tivoli Directory Server Web Administration Welcome panel.

Figure 17-10 shows the relevant screenshot.
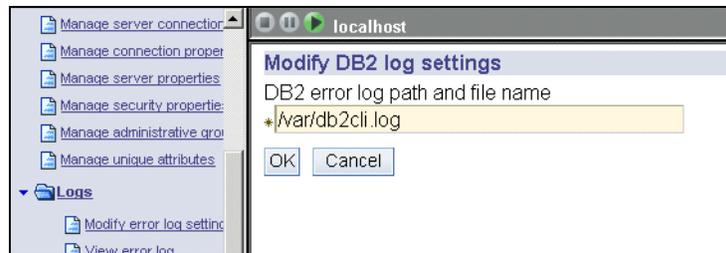
Figure 17-10   DB2 log settings

Using the command line

Issue the command:

```
ldapmodify -D <adminDN> -w <adminPW> -i <filename>
where <filename> contains:
    dn: cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas,
    cn=Configuration
    changetype: modify
    replace: ibm-slapdCLIErrors
    ibm-slapdCLIErrors: <newpathname>
```

To update the settings dynamically, issue the following **ldapexop** command:

```
ldapexop -D <adminDN> -w <adminPW> -op readconfig -scope single
"cn=Directory,cn=RDBM Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration" ibm-slapdCLIErrors
```

The **ldapexop** command updates only those attributes that are dynamic. For other changes to take effect you must restart the server. See Chapter 10, "Client tools" on page 237, to see which attributes can be updated dynamically.

Viewing the DB2 error log

Use the following procedures to view the DB2 error log.

Using Web Administration

To do this:

1. Expand **Logs** in the navigation area, then click **View DB2 log**.

2. The panel displays the first page of the DB2 log and the navigation arrows at the bottom of the panel enable you to go to the Next page or to the Previous page. From the menu, you can select a specific page, for example Page 6 of 16, and click **Go** to display that page of the DB2 log. You can:

   a. Click **Refresh** to update the entries in the log.

   b. Click **Clear log** to delete all entries in the DB2 error log.

   c. Click **Close** to return to the IBM Tivoli Directory Server Web Administration Welcome panel.

Figure 17-11 shows the relevant screen shot.
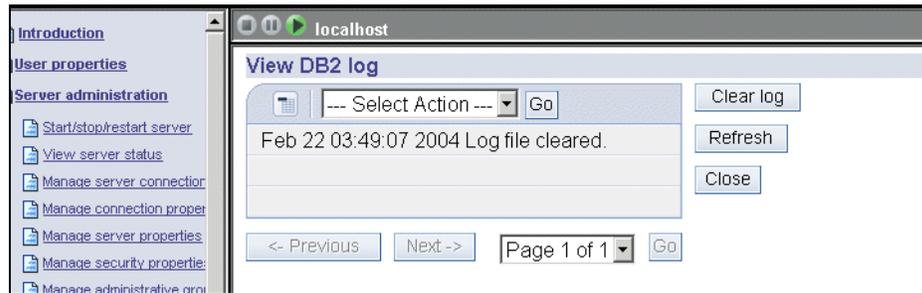
Figure 17-11   DB2 log contents

Using the command line

To view the DB2 error log issue the following command (on UNIX):

```
more /var/ldap/db2cli.log
```

Where var/ldap/db2cli.log is the default path for the DB2 error log.

> **Note:** /var/ldap/db2cli.log is the default DB2 error log for UNIX systems and ldapinstalldir\var\db2cli.log is the default DB2 error log for Windows systems.

To view and clear the DB2 error log dynamically:

```
ldapexop -D <adminDN> -w <adminPW> -op readlog -log cli -lines all
ldapexop -D <adminDN> -w <adminPW> -op clearlog -log cli
```

Here is an example of example of the above commands:

```
C:\>ldapexop -D <adminDN> -w <adminPW> -op readlog -log cli -lines all
     02/03/2004 10:24:57 PM native retcode = -601; state = "42710";
     message = "[IBM][CLI Driver][DB2/NT] SQL0601N  The name of the object
     to be created is identical to the existing name "LDAPBP" of type
     "BUFFERPOOL".  SQLSTATE=42710               "
```

```
C:\>ldapexop -D <adminDN> -w <adminPW> -op clearlog -log cli
       cli log file cleared.
C:\>ldapexop -D <adminDN> -w <adminPW> -op readlog -log cli -lines all
       Feb 22 03:49:07 2004 Log file cleared.
```

> **Note:** In case it is necessary to dig into DB2 errors further, you can go
> through a file known as db2diag.log. On UNIX, the default path of
> db2diag.log is: "<directory where you configured you're ldap
> database>/sqllib/sqldump/db2diag.log". On Windows, the default path is
> "<DB2 Installation path>\<You're DB2 instance name>\db2diag.log". For
> example, "D:\Program files\IBM\SQLLIB\LDAPDB2\db2diag.log". You can
> change the default path using DB2 utilities. Refer to Chapter 20,
> "Developing JNDI-based applications" on page 619, for further information
> on db2diag.log.

17.3 Operating system commands for monitoring ITDS

Sometimes it is required to track the resources consumed by the directory server
while running for long durations. Listed below are some OS-specific commands
to achieve the above goal.

AIX
To view information about the running process ibmslapd, issue the following
command:

```
ps auwx | grep -i ibmslapd
```

Linux
Command line tool to view information about the running process ibmslapd:

```
ps aux | grep -i ibmslapd
```

Graphical tool to view information about the running processes:

```
pstree (check the man pages for more details)
```

Solaris
Command line tool to view information about the running process ibmslapd:

```
ps -yel | grep -i ibmslapd
```

Graphical tool to view information about the running processes:

```
/usr/dt/bin/sdtprocess
```

You can specify the refresh rate at which the screen will be refreshed to show the updated statistics.

Windows

Command line utility to view information about running processes: Download the **pview** utility from the Microsoft Web site.

Graphical tool: The Processes tab in the Windows Task Manager can be used for monitoring the resource usage.

HP-UX

Command line utility to view information about the running ibmslapd process:

```
ps -eaf | grep -i ibmslapd
```

All the above commands, shown for different operating systems, help us to get to know two things.

▶ Firstly, if ibmslapd is still running.

▶ Secondly, how much has the process size grown till date and if it is within permissible limits or going to click the limits soon.

If there is growth in process size, it is not necessarily a memory leak. For example, if you have set your caches too large then as the number of misses on the directory server cache increases the data cached increases, which in turn increases the ibmslapd process size. The ulimits of the system (UNIX) play a very significant role here, in order to regulate the systems' resource utilization. You can see the current ulimit settings using the command:

```
ulimit -a
```

Here is a sample output of the ulimits of a system:

```
bash-2.05a# ulimit -a
core file size          (blocks, -c) 1048575
data seg size           (kbytes, -d) 131072
file size               (blocks, -f) 1048575
max memory size         (kbytes, -m) 32768
open files                      (-n) 2000
pipe size            (512 bytes, -p) 64
stack size              (kbytes, -s) 32768
cpu time               (seconds, -t) unlimited
max user processes              (-u) 262144
virtual memory          (kbytes, -v) unlimited
```

As seen above the (u)pper limit for the process memory size is 32 MB.

If you have set the process memory size to unlimited then ibmslapd would keep growing, till either the clients are happy with the entries in the cache or the entire directory has been cached. If neither case satisfies, ibmslapd would keep growing, ultimately bringing down the entire system to a hang condition. The only alternative to do in this situation would be to reboot the system physically. In order to avoid such issues, please ensure:

► You have set you're systems' ulimits appropriately.

► You have the LDAP caches set appropriately.

► You have DB2 bufferpools set appropriately.

By appropriately, we mean as per the availability of resources, so that if the ibmslapd size grows beyond this extent, the OS will just pull out ibmslapd out of the process table, that is, it'll kill ibmslapd on its own.

Such things are most commonly observed while tuning the directory server to get to know the answers to a set of performance queries. Queries like "What exact figures of the caches will suit my environment?", "What are the database bufferpools we are supposed to set my systems to?", "What are the attributes that we want to cache", "What should be the size of my attribute cache?" etc. These are the types of tests where we need to have an eye on the size of the ibmslapd process.

If anything unusual happens, either you have overshot one of you're parameters or there might be a genuine memory leak, which needs to be brought forth the ITDS Support team.

Here is an example of the `ps auwx` command on AIX:

```
bash-2.05a# ps auwx | head -1
USER        PID %CPU %MEM   SZ  RSS    TTY STAT    STIME  TIME COMMAND
bash-2.05a# ps auwx | grep ibmslapd | grep -v grep
ldap     340136  0.2  1.0 13332 13444     - A       Feb 20 30:14 ibmslapd
```

As seen above the current size of ibmslapd seems to be: 13 MB, which is very well within limits. Make sure the size of ibmslapd does not show anomalous growth in you're environment.

Well, we have seen a lot of things in this chapter, which help in monitoring our directory server. Prior to summarizing things, let us go over a couple more notes which are worth having a look.

> **Note:** See the following:
>
> ► In the Web Administration Tool the Logfiles field in the task title bar accesses the Web Administration console log files. The IBM Tivoli Directory Server log files are accessible by using the procedures specified in the sections, we have just discussed.
>
> ► On Windows-based systems, if a path begins with the drive letter and a colon, it is assumed to be the full path. A path without the drive letter, starts in the installation tree. As examples: c:\tmp\mylog is a full path, while \tmp\mylog is interpreted as c:\program files\ibm\ldap\tmp\mylog.
>
> ► The simplest way to get to a problem is to know the time when it has occurred. The log files are timestamped. So you just compare the different log files simultaneously for the activities at a given instant of time and there you are, very close to the problem cause. If multiple LDAP servers are involved (for example, debugging a replication issue), keeping them time synchronized is handy (Only of course if time synchronization if feasible).

17.4 Summary

In summary:

► To start with the chapter, we looked into the reasons for monitoring our directory server.

► Then we looked into the different monitoring tools:

– Client tools to monitor the directory server, whereby we saw that the search to "cn=monitor" provides a lot of insight into the directory performance.

– Log files helped us in knowing if there were any configuration issues that need to be overcome for the smooth functioning of the database.

– A separate database which helped us in knowing what all changes took place with the directory server at different instants of time.

► Then we saw how the OS utilities help in tracking the anomalies associated with the directory server growth.

Part 4

Developing directory-enabled applications

As seen throughout this book, many applications are already LDAP-enabled. They utilize LDAP directories for various purposes, user information, authentication/authorization, configuration settings, and so forth. User applications can benefit from the advantages of directories as well. This chapter gives ideas on how to leverage LDAP directories in self-written applications and introduces the various programming interfaces and methods to directory-enable applications.

For example, a company sets up an enterprise LDAP directory for e-mail clients to use to retrieve e-mail addresses. The information in the directory can be used for various other purposes. The payroll application, for example, could use the directory to retrieve employee addresses. Employees may also want to update their own information in the directory. All of these uses require some sort of programming unless the company has bought software that does exactly what is

needed. Virtually all applications whether they are written in C++, Java, Visual Basic, etc., can be LDAP-enabled due to the variety of different application interfaces available.

IBM Directory Server provides a set of Application Programming Interfaces (APIs) that allow users to search a directory or perform operations, such as additions, modifications, or deletions of directory entries. This part of the book contains some examples of how to use APIs in a C or Java application to search for a specific directory entry or to add a new entry into a directory.

18

Debugging IBM Tivoli Directory Server related issues

This chapter discusses the various debugging and tracing capabilities provided by ITDS 5.2. These facilities provide a directory administrator greater insight into what on side of the directory any given time.

18.1 Overview

The process of installing and configuring ITDS, for various reasons, is not always error free. The directory administrator can encounter problems with basic installation of the server, configuring various server components, or the server might fail to start for no obvious reason.

Debugging is the process of finding the cause of the problem using various tools and techniques and eliminating them. Due of these inherent problems, ITDS provides administrators various command line options, tools and detailed log files that help the user find the cause of the problem.

18.2 Debugging problems

The following sections describe how to debug configuration problems, directory server errors, directory server debug modes, and DB2 debug logs.

18.2.1 Debugging configuration problems

The first thing that is to be done after installation of the product is the configuration. If this fails then there is no way you can go but to resolve the issue. The very basic steps towards making the IBM Tivoli Directory Server up and running are:

► Configuring an Admin DN and password

► Configuring the Directory server database

The above basic tasks are performed by using the configuration tools provided by the directory server. These are:

► ldapcfg: Command-line tool for configuring the directory server (admin dn, password, database and other tasks).

► ldapxcfg: GUI for doing the same tasks as ldapcfg. In other words, it is the GUI equivalent of ldapcfg.

► ldapucfg: Command-line tool for unconfiguring the directory server(admin dn, password, directory and changelog database etc.).

While the configuration of Admin DN and password is fairly straightforward and is less error prone, the database configuration may not be that easy. Generally, the only reasons the configuration of the Admin DN fails are if the IBM Directory configuration file (<ldapinstalldir>/etc/ibmslapd.conf) has had the permissions accidentally changed, or if the user enters an invalid DN. If the database

configuration fails, the following sources can be checked to find the cause of the failure:

1. Output on the screen

   All of the configuration programs are either started from a console command line prompt (ldapcfg, ldapucfg) or open a background console (ldapxcfg). As the database configuration progresses, status messages (and limited error messages) are displayed in the associated console window. If a problem occurs, the user should copy these messages to the system clipboard and then save them in a file for the support teams.

2. DB2 log files

   If the error is a direct error from DB2, then DB2 often creates message/error files in the /tmp directory (on UNIX platforms). If the user has a database configuration problem on UNIX systems, they need to examine all of the files in the /tmp directory that were created around the time of the attempted configuration. On Windows systems, examine any DB2 error logs in the directory named for the instance you were trying to configure under the DB2 install directory under. For example, if your were trying to create the default ldapdb2 instance and database, and if your DB2 was installed in D:\sqllib, then you need to examine the files in the D:\sqllib\ldapdb2 directory if it exists. Especially look for and examine the db2diag.log file in that directory.

3. IBM Directory logs for configuration issues

   IBM Directory logs most configuration errors in the file ldacfg.out. On UNIX platforms, this file can be found in the /tmp directory. On Windows platforms, this file is created in the root directory of the drive you ran configuration from.

If the above sources are not sufficient for determining the cause of the problem, we can resort to advanced debugging. In advanced debugging we set two environment variables and collect the relevant logs as explained below:

1. JAVA_DEBUG

   Set this environment variable to any non-empty value, for example:

   `JAVA_DEBUG=1`

   On UNIX platforms, use export JAVA_DEBUG=1. This causes certain Java debug information built into the code to be displayed on stdout (the console). The best practice is to redirect the output to a file and then analyses the same.

2. LDAP_DBG

   Set this environment variable to any non-empty value. For example:

   `LDAP_DBG =1`

On UNIX platforms, use export LDAP_DBG=1. This causes a debug file to be created for the IBM support and development teams. The file name that is created is dbg.log. It is created in /var/ldap/ directory.

On the Windows NT and Windows 2000 platforms, dbg.log is created in the <ldapinstalldir> \var directory.

18.2.2 Debugging directory server related errors using log files

When a problem occurs that appears to be related to the IBM Directory Server, you should first check the following files for error messages. The default locations of these files are /var/ldap in case of Unix and <ldapinstalldir>\var in case of windows.

► ibmslapd.log: When the server starts up, it logs all messages to this file. During the normal operations of the directory server, if any operation is requested of the server, which it does not like to do, it would log the same. For instance, if a user tries to add an object with an invalid schema, server will not add the same and would record the relevant message in ibmslapd.log. Moreover, you can also get information like the number of additions that ldif2db did, or the number of entries that db2ldif managed to export successfully etc. Hence, ibmslapd.log is quite handy in resolving issues.

► audit.log: The audit log shows what searches are being performed and the parameters used in each search. The audit log also shows the timestamp of when a client binds and unbinds from the directory, for that matter it timestamps all the operations it is supposed to log. Observing these measurements allows the LDAP administrator to identify LDAP operations that take a long time to complete.

► db2cli.log: All errors encountered while the directory server tries to access the backend database using CLI (Call Level Interface) routines are logged into the db2cli.log file.

You can change the location of these files by modifying the appropriate settings from the Web administration tool or by directly editing the ibmslapd.conf file.

For more information on how to handle the above log files, please go through Chapter 17, "Monitoring IBM Tivoli Directory Server" on page 547.

18.2.3 Using server debug modes

If the above listed log files do not provide enough information about the problem, you can run the IBM Tivoli Directory Server in a special debug mode that generates very detailed information. An ibmslapd trace provides a list of the SQL commands issued to the DB2 database. These commands can help you identify operations that are taking a long time to complete.

Here is a snippet of the server debug trace:

```
053:08:12:32 T1452    61882676          423 usec SQLExecute => 0, hstmt=20001
053:08:12:32 T1452    61882977           26 usec SQLFetch => 100, hstmt=20001
```

The above statements show that the time taken to do the SQLExecutes of the statement with a handle of 20001 is 423 micro seconds and that the time required by the SQLFetch for the same SQL statement is 26 micro seconds.

In some cases, the log files you have collected seem to provide that some problem had occurred with the directory server, but fail to hint at the probable cause. The debug trace would help in such situations, as it would give more detailed steps of the server operations. The best example would be, suppose due to some error, the database starts up in configuration mode. The logs will not provide you the problem cause. They will just hint that there was a problem with the server. So how do you get to the root of the problem? Of course yes, by running the server in debug mode. That helps out to know the exact cause of the server starting up in configuration mode. You just correct the problem, pointed to, by the trace, and there you are, you're server is up and running once again.

The server executable ibmslapd must be run from a command prompt to enable debug output. The syntax is as follows:

```
ldtrc on -l 20000000
ibmslapd -h <bitmask>
```

The above means to turn (on) the ldap trace and keep a buffer of 20 Million lines. that is, out of the total trace information only the last 20 Million lines would be kept in the buffer and not more than that. This is handy sometimes, in situations where you want to reduce the size of the debug file to be analyses and are sure that you would stop the server instantly when the problem occurs. Let us see what the above commands mean. First We will look into what ldtrc means for us.

ldtrc

This command is used for enabling/disabling the trace options for ibmslapd. In the sense that you would need to use this command to turn on the trace options for allowing ibmslapd to run in debug mode. What debug level ibmslapd runs at is a later issue. This command is just a preparatory step towards enabling the debug mode of ibmslapd.

ldtrc usage:

```
Usage: ldtrc (chg|clr|dmp|flw|fmt|inf|off|on) options
   chg|change : change the trace mask, pid.tid, cpid or maxSevereErrors
   clr|clear : clear the trace
   dmp|dump : dump the trace to a binary trace file
   flw|flow : show control flow of the trace
   fmt|format : format the trace
```

```
    inf|info|information : get information on the trace
    off : turn the trace off
    on : turn the trace on
For more information type ldtrc (chg|clr|dmp|flw|fmt|inf|off|on) help
```

The further details for the above shall not be given here and can be found in the description of ldaptrace in one of the following sections.

ibmslapd in debug mode

Once you have turned on the trace options using ldtrc, we can go ahead with collecting the server debug information by running ibmslapd in debug mode. The server is run in debug mode by specifying a bitmask. The bitmask helps the server in deciding the set of operations it is supposed to run with extensive tracing. Like if you have replication related issues and want to debug the replication operations of the server, you can turn on just the flags pertaining to replication (1024) that is, specify this debug bitmask while starting the server. A value of 65535 for the bitmask indicates that the maximum debug output should be generated. Table 18-1 describes the various flags you can set in the bitmask.

Table 18-1   Debug categories

Hex	Decimal	Value	Description
0x0001	1	LDAP_DEBUG_TRACE	Entry and exit from routines
0x0002	2	LDAP_DEBUG_PACKETS	Packet activity
0x0004	4	LDAP_DEBUG_ARGS	Data arguments from requests
0x0008	8	LDAP_DEBUG_CONNS	Connection activity
0x0010	16	LDAP_DEBUG_BER	Encoding and decoding of data
0x0020	32	LDAP_DEBUG_FILTER	Search filters
0x0040	64	LDAP_DEBUG_MESSAGE	Messaging subsystem activities and events
0x0080	128	LDAP_DEBUG_ACL	Access Control List activities
0x0100	256	LDAP_DEBUG_STATS	Operational statistics
0x0200	512	LDAP_DEBUG_THREAD	Threading statistics
0x0400	1024	LDAP_DEBUG_REPL	Replication statistics
0x0800	2048	LDAP_DEBUG_PARSE	Parsing activities
0x1000	4096	LDAP_DEBUG_PERFORMANCE	Relational backend performance statistics

Hex	Decimal	Value	Description
0x2000	8192	LDAP_DEBUG_RDBM	Relational backend activities (RDBM)
0x4000	16384	LDAP_DEBUG_REFERRAL	Referral activities
0x8000	32768	LDAP_DEBUG_ERROR	Error conditions
0xffff	65535	LDAP_DEBUG_ANY	All levels of debug

Now depending upon the type of operations you want to debug, just form the relevant bitmask by ORing the individual bitmasks and pass the consolidated bitmask to ibmslapd, during its startup.

The trace output will be directed to the standard output. It is recommended that you redirect the same to a file for analyzing later, as the console buffer might not be sufficient to retain all the information on the screen. The file redirection can be achieved by following any of the given steps below.

For Windows:

```
ibmslapd -h bitmask > filename 2>&1
OR
set LDAP_DBG_FILE=filename
ibmslapd -h <bitmask> 2>&1
```

For Unix:

```
ibmslapd -h bitmask 2&>1 | tee filename
OR
export LDAP_DBG_FILE=filename
ibmslapd -h <bitmask> 2>&1
```

Running a trace on several operations will definitely result in slow performance, so remember to turn the trace off when you are finished using it by the following command:

```
ldtrc off
```

It is not always feasible to have the server run in debug mode for all times especially in a production environment, where high performance is always a mandate. In such circumstances, it is recommended to go for dynamic tracing, as will be seen in the next section.

Dynamic tracing

Sometimes it is observed that there are problems with the directory server, after running the directory server for a long time that is, say of the order of eight hours. The log files indicate that there is some problem, but it is not clear. However, it is essential to get to know the problem, because that's impacting business. What

do we do in such cases? It is not feasible to run the server in debug mode for say 8 hours and then analyze the tons of data that this will generate, taking a lot more time in debugging the issue. In some cases, we aren't even sure that the problem would get generated after taking all the efforts of running the server in debug mode, in a planned downtime. Such situations are common and in such circumstances, it is better to go for dynamic tracing of the server. That would not impact the server's performance, during the period when we know that the problem will not occur and also the amount of trace generated would be comparatively much smaller to amount generated using the static tracing. The utility being used for dynamic tracing is known as ldaptrace. Let us study this tool in more depth.

ldaptrace

The administration tracing utility, ldaptrace, is used to dynamically activate or deactivate tracing of the Directory Server. This extended operation can also be used to set the message level and specify the name of the file, where the output is written. If LDAP trace facility (ldtrc) options are requested, they must be preceded by --.

To display syntax help for ldaptrace, type:

```
ldaptrace -?
```

Note: While the ldaptrace utility can be used with SSL or TLS, only the simple bind mechanism is supported.

Here is a synopsis of what sort of parameters ldaptrace would take:

```
ldaptrace -a port -l [on|off|clr|chg|info|dump] --[ldtrc options] -D
adminDn -h hostname -K keyfile -m debugLevel -N key_name -o debugFile -p
port -P key_pw -t [start|stop] -v -w adminPW -Z -?
```

Note: Only the administrator or a member of the administrative group can use this utility.

Using ldaptrace consumes resources and affects the performance of the server.

Options of ldaptrace

The options are:

► -a *port*: Specifies an alternate TCP port where IBM Administration Daemon (ibmdiradm), not the Directory Server, is listening. The default port is 3538. If not specified and -Z is specified, the default SSL port 3539 is used.

► -l *[on|off|clr|chg|info|dump] -[ldtrc options]*:

– on: Turns on the tracing facility. You can specify any of the following ldtrc options preceded by an extra -.

• [-m <mask>] where <mask> = <products>.<events>.<components>.<classes>.<functions>.

• [-p <pid>[.<tid>]] traces only the specified process or thread.

• [-c <cpid>] traces only the specified companion process.

• [-e <maxSevereErrors>] stops tracing after the maximum number of severe errors (maxSevereErrors) is reached.

• [-s | -f <fileName>] sends the output to shared memory or a file.

• [-l [<bufferSize>] | -i [<bufferSize>]] specifies to retain the last or the initial records. The default buffer is 1M.

• [-this <thisPointer>] trace only the specified object.

> **Note:** The tracing facility must be on for server data to be traced.

– off: Turns off the tracing facility.

– clr: Clears the existing trace buffer.

– chg: The trace must be active before you can use the chg option to change the values for the following ldtrc options:

• [-m <mask>] where <mask> = <products>.<events>.<components>.<classes>.<functions>.

• [-p <pid>[.<tid>]] traces only the specified process or thread.

• [-c <cpid>] traces only the specified companion process.

• [-e <maxSevereErrors>] stops tracing after the maximum number of severe errors (maxSevereErrors) is reached.

• [-this <thisPointer>] trace only the specified object.

– info: Gets information about the trace. You must specify the source file which can be either a binary trace file, or trace buffer and a destination file. The following is an example of the information that the info parameter gives:

```
C:\>ldtrc info
Trace Version : 1.00
Op. System : NT
Op. Sys. Version : 4.0
H/W Platform : 80x86
Mask : *.*.*.*.*.*
```

```
pid.tid to trace : all
cpid to trace : all
this pointer to trace : all
Treat this rc as sys err: none
Max severe errors : 1
Max record size : 32768 bytes
Trace destination : shared memory
Records to keep : last
Trace buffer size : 1048576 bytes
Trace data pointer check: no
```

– dump: Dumps the trace information to a file. This information includes process flow data as well as server debug messages. You can specify the name of the destination file where you want to dump the trace. The default destination files is:

```
For Unix-based systems:
/var/ldap/ibmslapd.drace.dump
For Windows-based systems:
<installationpath>\var\ibmslapd.trace.dump
```

> **Note:** This file contains binary ldtrc data that must be formatted with the ldtrc format command.

► -h *ldaphost*: Specify an alternate host on which the Directory Server and the Administration Daemon are running.

► -K *keyfile*: Specify the name of the SSL or TLS key database file with default extension of kdb. If the key database file is not in the current directory, specify the fully-qualified key database filename. If a key database filename is not specified, this utility will first look for the presence of the SSL_KEYRING environment variable with an associated filename. If the SSL_KEYRING environment variable is not defined, the default keyring file will be used, if present.

A default keyring file that is, ldapkey.kdb, and the associated password stash file that is, ldapkey.sth, are installed in the /lib directory under LDAPHOME, where LDAPHOME is the path to the installed LDAP support. LDAPHOME varies by operating system platform:

► AIX operating systems - /usr/ldap
► HP-UX operating systems - /usr/IBMldap
► Linux operating systems - /usr/ldap
► Solaris operating systems - /opt/IBMldaps
► Windows operating systems - c:\Program Files\IBM\LDAP

> **Note:** This is the default install location. The actual LDAPHOME is determined during installation.

See *IBM Directory C-Client SDK Programming Reference* for more information about default key database files, and default Certificate Authorities. This document can be found at:

`http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html`

If a keyring database file cannot be located, a "hard-coded" set of default trusted certificate authority roots is used. The key database file typically contains one or more certificates of certificate authorities (CAs) that are trusted by the client. These types of X.509 certificates are also known as trusted roots. For more information on managing an SSL or TLS key database, see "SSL/TLS support" on page 455.

This parameter effectively enables the -Z switch.

► -m *debuglevel*: Set the mask debugging level for server debug messages. Refer Table 20-1 on page 622 for more information on the debuglevel to be used. It is same as the bitmask, we pass to ibmslapd, as already discussed.

► -N *certificatename*: Specify the label associated with the client certificate in the key database file. If the LDAP server is configured to perform server authentication only, a client certificate is not required. If the LDAP server is configured to perform client and server Authentication, a client certificate might be required. Certificatename is not required if a default certificate/private key pair has been designated as the default. Similarly, certificatename is not required if there is a single certificate/private key pair in the designated key database file. This parameter is ignored if neither -Z nor -K is specified.

► -o *debugfile*: Specifies the output file name for the server debug messages.

► -p *port*: Specify an alternate TCP port where the ldap server is listening. The default LDAP port is 389. If not specified and -Z is specified, the default LDAP SSL port 636 is used.

► -P *keyfilepw*: Specify the key database password. This password is required to access the encrypted information in the key database file, which may include one or more private keys. If a password stash file is associated with the key database file, the password is obtained from the password stash file, and the -P parameter is not required. This parameter is ignored if neither -Z nor -K is specified.

► -t *[start|stop]*:

  – *start* starts the collection of server trace data.
  – *stop* stops the collection of server trace data.

► -v: Specifies to run in verbose mode.

- ► -w *adminPW* | *?* : Use adminPW as the password for authentication. Use the ? to generate a password prompt. Using this prompt prevents your password from being visible through the ps command.

- ► -?: Displays the help screen.

Now let us see some examples for using the ldaptrace utility.

To turn the ldtrc facility on and start the server trace with a 2M trace buffer, issue the command:

```
ldaptrace -h <hostname> -D <adminDN> -w <adminpw> -l on -t start --l
2000000
```

To stop the server trace, issue the command:

```
ldaptrace -h <hostname> -D <adminDN> -w <adminpw> -t stop
```

To turn off the ldtrc facility, issue the command:

```
ldaptrace -h <hostname> -D <adminDN> -w <adminpw> -l off
```

To collect the trace over SSL, issue the command:

```
ldaptrace -h <hostname> -D <adminDN> -w <adminPW> -Z -K <SSL key db> -P
<SSL key db password> -l on -t start --l 2000000
```

Thus using ldaptrace we can debug the server dynamically. Consequently we can maintain the performance of the server, by not running it in debug mode all the time and by taking the desired debug information as and when necessary.

18.2.4 DB2 error log file

The main db2 log file you need to check in case of errors is db2diag.log. This file is mainly meant for the support people and the developers to analyze.

In case of UNIX systems, it is located in <Directory containing your database>/sqllib/db2dump directory. In windows it is located in x:\sqllib\<instance_name> directory where x is the drive where DB2 is installed.

Note: The location of the db2diag.log file is controlled by the DB2 server configuration parameter DIAGPATH, so the directory paths on your system might be different from the default paths.

You can control the level of detailed information that is written to the db2diag.log file by using the DIAGLEVEL configuration parameter and the DLFM_LOG_LEVEL registry value.

DIAGLEVEL

Determines the severity of DB2 diagnostic information recorded in the db2diag.log error log file. Valid values are from 1–4. 1 denotes that a minimal amount of information is to be recorded, and 4 denotes that the maximum amount of information is to be recorded. The default setting is 3. You can increase the amount of error information recorded using the following command: `db2 update dbm cfg using DIAGLEVEL 4`. This setting should be changed only at the request of IBM service or development for debugging purposes.

DLFM_LOG_LEVEL

Determines the severity of DLFM diagnostic information recorded in the db2diag.log error log file. Its default setting is LOG_ERR. You can increase the amount of error information recorded using the following command:

```
db2set DLFM_LOG_LEVEL=LOG_DEBUG
```

18.3 Summary

In summary:

► To start with, we went through the necessity of debugging the server.

► Then we went through different types of the problems encountered with the directory server and the means to debug the same.

   – Configuration problems
   – Server related problems
      • Using log files
      • Using the server debug trace
      • Using the dynamic trace facility to debug server operations

► Finally we went through debugging issues pertaining to DB2.

19

Developing C-based applications

Many C-based applications will want to make use of directory based information. The IBM Directory Server C-Client SDK includes various sample LDAP client programs, and an LDAP client library used to provide application access to the LDAP servers. In this chapter, sample code is provided to connect and search a directory and get the results. In addition sample code is provided to modify a directory entry. More information about the C-Client SDK can be found at:

http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html

19.1 Overview

Whether writing new C-based applications or modernizing existing applications, there are many benefits from directory-enabling them. The IBM Directory Server C-Client SDK provides a rich set of application programming interfaces (APIs) that allow developers to search and update entries in an LDAP directory. This chapter gives examples of how to use some of these APIs for searching and updating the directory.

The set of LDAP APIs are designed to provide a suite of functions that can be used to develop directory-enabled applications. Directory enabled applications will typically connect to one or more directories and perform various directory-related operations, such as:

► Performing binds
► Adding entries
► Searching the directory and obtaining the resulting list of entries
► Deleting entries
► Modifying entries
► Renaming entries
► Setting LDAP controls

The type of information that is managed in the directory depends on the nature of the application. Directories are often used to provide public access to information about people, including:

► Name information
► Phone numbers
► E-mail addresses
► Fax numbers
► Mailing addresses

Increasingly, directories are being used to manage and publish other types of information, including:

► Configuration information
► Public key certificates (managed by Certification Authorities)
► Access control information
► Locating information (how to find a service)

The LDAP APIs provide for both synchronous and asynchronous access to a directory. Asynchronous access makes it easy for the application to do other work while waiting for the results of a potentially lengthy directory operation to be returned by the server.

Source code, example makefile, and executable programs are provided with the IBM Directory Server Client SDK for performing the following operations:

- ► `ldapchangepwd` - changes a user's password
- ► `ldapsearch` - searches the directory
- ► `ldapmodify` - modifies information in the directory
- ► `ldapdelete` - deletes information from the directory
- ► `ldapmodrdn` - modifies the Relative Distinguished Name (RDN) of an entry in the directory

19.2 Typical API usage

The basic interaction is as follows:

1. A connection is made to an LDAP server by calling either `ldap_init` or `ldap_ssl_init`, which is used to establish a secure connection over Secure Sockets Layer (SSL).

2. An LDAP bind operation is performed by calling `ldap_simple_bind`. The bind operation is used to authenticate to the directory server. Note that the LDAP V3 API and protocol permits the bind to be skipped, in which case the access rights associated with anonymous access are obtained.

3. Other operations are performed by calling one of the synchronous or asynchronous routines (for example, `ldap_search_s` or `ldap_search` followed by `ldap_result`).

4. Results returned from these routines are interpreted by calling the LDAP parsing routines, which include operations such as:

   - `ldap_first_entry`, `ldap_next_entry`
   - `ldap_get_dn`
   - `ldap_first_attribute`, `ldap_next_attribute`
   - `ldap_get_values`
   - `ldap_parse_result` (new for LDAP V3)

5. The LDAP connection is terminated by calling `ldap_unbind`.

When handling a client referral to another server, the `ldap_set_rebind_proc` routine defines the entry point of a routine called when an LDAP bind operation is needed.

For more detailed information on the API calls mentioned above, please refer to the *IBM Tivoli Directory Server 5.2 C-Client SDK Programming Reference Guide*. This guide is available at:

`http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html`

19.3 API flow when searching a directory

This section provides an overview of what APIs can be used to perform a search operation on a LDAP directory. This example, based on LDAP Version 3 APIs, shows one way of searching the directory using a non-secure session in synchronous mode. There are also APIs available to initiate an SSL session to the LDAP server. The APIs are documented in the order they have to be used. Figure 19-1 shows an overview of the APIs and the order in which they are used.
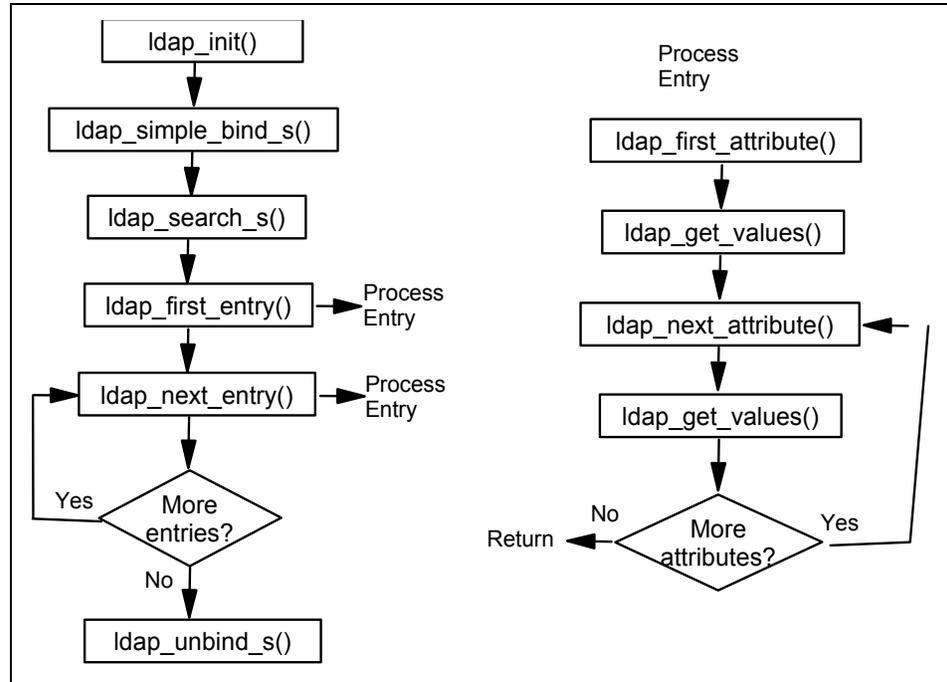
Figure 19-1   Overview of APIs used for searching a directory

In the example given in Figure 19-1 the APIs are processed in a certain order. The following information explains the purpose of each API used in the example.

19.3.1 ldap_init()

The `ldap_init()` API initializes a session with an LDAP server. The server is not actually contacted until an operation is performed that requires it, allowing various options to be set after initialization, but before actually contacting the host. It allocates an LDAP structure that is used to identify the connection and maintain per-connection information. The input parameters required are the host and port number of the LDAP server. The `ldap_init()` function returns a pointer

to an LDAP structure, which should be passed to subsequent calls to other LDAP functions such as `ldap_simple_bind_s()` and `ldap_search_s()`.

19.3.2 ldap_simple_bind_s()

The `ldap_simple_bind_s()` function is used to authenticate a distinguished name (DN) to a directory server. There are other APIs available to authenticate users with a different authentication method. With LDAP Version 3 the bind API can be skipped allowing an anonymous connection to the directory server. However, anonymous access will have limited access on the majority of LDAP servers. The `ldap_simple_bind_s()` API requires as input parameters: the LDAP structure ld as returned by the `ldap_init()` API, the distinguished name (DN) of the entry performing the bind, and the password. The DN used has to have the authorities to perform the intended changes. The return code of this API indicates a successful bind or another error code.

19.3.3 ldap_search_s()

The `ldap_search_s()` API is used to perform an LDAP search operation. `ldap_search_s()` is a synchronous request. This API requires the LDAP structure that was returned by the `ldap_init()` API as an input parameter. The remaining input parameters define the search base, scope of search, search filter, attributes to be returned, and whether to return only attribute names or names as well as values. Entries returned from the search (if any) are contained in the res parameter. When an LDAP operation completes and the result is obtained as described, a list of LDAPMessage structures is returned. This is referred to as the search result chain. A pointer to the first of these structures is returned by `ldap_search_s()` API. However, the results cannot be used in the form returned. They have to be parsed by the corresponding APIs to process the returned entries, their attributes, and the attribute values as depicted in Figure 19-1 on page 606.

19.3.4 ldap_first_entry()

In the search example, this API is used to parse results for the first entry received from the synchronous LDAP search function `ldap_search_s()`. Used an input parameters are the LDAP structure that was returned by the `ldap_init()` API and the result LDAPMessage structure returned by the `ldap_search_s()` API. The latter value is the pointer to the first entry returned by the search function. The return value is the pointer to the first entry of the search results and is required as an input parameter for the `ldap_first_attribute()` API.

19.3.5 ldap_first_attribute()

The `ldap_first_attribute()` API returns the first attribute in an entry.
`ldap_first_attribute()` takes the LDAP structure returned by the `ldap_init()`
API and an entry returned by `ldap_first_entry()` or `ldap_next_entry()`. In
addition it has an output parameter that contains a pointer to an opaque data
structure for data encoded with Basic Encoding Rules (BER). This pointer is
used in subsequent calls to the `ldap_next_attribute()` API to keep track of the
current position. It returns a pointer to a buffer containing the first attribute type in
the entry.

19.3.6 ldap_get_values()

The `ldap_get_values()` API is used to retrieve attribute values from an LDAP
entry as returned by `ldap_first_entry()` or `ldap_next_entry()`. The input
parameters for the `ldap_get_values()` API are the pointer to the entry as
returned by the `ldap_first_entry()` or `ldap_next_entry()` APIs and the pointer
to the buffer containing the attribute type as returned by the
`ldap_first_attribute()` or `ldap_next_attribute()` APIs. The
`ldap_get_values()` returns a NULL-terminated array of the attribute's values.
Remember that an attribute value can contain more than one value.

19.3.7 ldap_next_attribute()

Once the values of the first attribute have been processed, a loop can be used to
process the remaining attributes of the current entry. The
`ldap_next_attribute()` API takes the LDAP structure returned by the
`ldap_init()` API and the entry returned by `ldap_first_entry()` or
`ldap_next_entry()`. In addition it has an input/output parameter that contains the
pointer that is used to keep track of the current position. For the first time the
`ldap_next_atttribute()` API is called, the pointer is the one returned by the
`ldap_first_attribute()` API. It returns a pointer to a buffer containing the next
attribute type in the entry. Processing continues with the `ldap_get_values()` API
until a NULL value is received indicating that no more attributes are available in
the current entry.

19.3.8 ldap_get_values()

The `ldap_get_values()` API is now used to retrieve attribute values from the
subsequent attributes returned by the `ldap_next_attribute()` API.

19.3.9 ldap_next_entry()

After the attributes and values of the first entry have been processed, the next entry from the search results can be processed using the `ldap_next_entry()` API. The input parameters needed are the LDAP structure that was returned by the `ldap_init()` API. The second parameter is the pointer to the entry as returned by the `ldap_first_entry()` or for subsequent calls to the `ldap_next_entry()` API by the `ldap_next_entry()` API. The return value is the pointer to the next entry of the search results and is required as an input parameter for the `ldap_first_attribute()` and `ldap_next_attribute()` APIs. The next entries' attributes and their values can now be processed using the next entry as described in `ldap_first_attribute()`. A return value of NULL indicates that no more entries are in the search results to be processed.

19.3.10 ldap_unbind_s()

After all entries have been processed the application must unbind from the LDAP server using, as in this example, the `ldap_unbind_s()` API. The API is used to end the connection to the LDAP server and free the resources contained in the LDAP structure that was created by the `ldap_init()` API.

> **Note:** Several of the APIs mentioned in this section allocate memory and resources. It is strongly recommended to use APIs, such as `ldap_memfree()`, `ldap_msgfree()`, and `ldap_control_free()`, to free up the allocated resources.

19.4 Sample code to search a directory

The sample application shown in Example 19-1 was written to help developers understand the various tasks involved to use the API to search an LDAP-based directory. It was written to provide a proof of concept.

> **Important:** The sample application shown in Example 19-1 does not cover and act on all possible exceptions, nor is it fully tested under all possible circumstances. It is a working application that can be used as an example and be extended to build a complete application.

Example 19-1   Code to search a directory using the C API

```
/* c_search.c - generic program to display ldap search results to STDOUT */

#include <stdio.h>
#include <string.h>
```

```c
#include <stdlib.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <ldap.h>

/* global variables */
static char *binddn = "cn=root";
static char *bindpwd = "password";
static char *ldaphost = "serverA.ibm.com";
static int ldapport = LDAP_PORT;
static char *ldapbase = "o=ibm,c=us";
static int referrals = LDAP_OPT_ON;
static int deref = LDAP_DEREF_NEVER;
static int ldapversion = LDAP_VERSION3;

main( int argc, char **argv )
{
   int rc;          // return code
   int i = 0;    // counter
   const char *errormsg = NULL;// error msg
   LDAP *ld;     // Ldap Object
   LDAPMessage *searchResult;// LDAPMessage used to get searchResult
   LDAPMessage *ldapEntry;// LDAPMessage used to retrieve entries
   BerElement *ber;// BER element
   char *attr = NULL;// attribute pointer
   char **values = NULL;// values pointer

   /* open connection to server */
   if ((ld = ldap_init(ldaphost, ldapport)) == NULL)
   {
      perror("ldap_init");
      exit(1);
   }

   // BIND to server using userid and password
   rc = ldap_simple_bind_s( ld, binddn, bindpwd );

   // Check to make sure BIND was successful otherwise exit
   if ( rc != LDAP_SUCCESS )
   {
      errormsg = ldap_err2string(ldap_get_errno(ld));
      fprintf(stderr, "ldap_bind_s: %s \n", errormsg);
      exit(rc);
   }

   // Perform search for all user objects in directory
   rc = ldap_search_s(ld, ldapbase, LDAP_SCOPE_SUBTREE,
"(objectclass=inetorgperson)", NULL, 0, &searchResult);
```

```
// Check to ensure search was successful
if ( rc != LDAP_SUCCESS )
{
   errormsg = ldap_err2string(ldap_get_errno(ld));
   fprintf(stderr, "ldap_search_s: %s \n", errormsg);
   exit(rc);
}

// Get first entry from searchResult object
ldapEntry = ldap_first_entry(ld, searchResult);

// Continue to loop until we have no more entries
while (ldapEntry != NULL)
{
   // output dn to STDOUT
   printf("%s\n", ldap_get_dn(ld, ldapEntry));

   // Get first attribute
   attr = ldap_first_attribute(ld, ldapEntry, &ber);

   // Continue to loop as long as we still have attributes
   while ( attr != NULL)
   {
      // Get the array of values for the current attribute
      values = ldap_get_values(ld, ldapEntry, attr);

      i=0;
      // Enumerate thru the array until we have printed all values to screen
      while (values[i] != NULL)
      {
         printf("%s=%s\n", attr, values[i]);
         i++;
      }

      // Get the next attribute
      attr = ldap_next_attribute(ld, ldapEntry, ber);
   }

   // Get the next entry
   ldapEntry = ldap_next_entry(ld, ldapEntry);
   printf("\n");
}

// Clean up allocated memory
ldap_msgfree(searchResult);
ldap_ber_free(ber);
ldap_memfree(attr);
ldap_value_free(values);
```

```
    /* unbind and exit */
    ldap_unbind_s(ld);
    exit(rc);
}
```

19.5 API flow when updating a directory entry

This section provides an overview of what APIs can be used to perform an update of attributes for an existing entry in the LDAP directory. This example, based on LDAP Version 3 APIs, shows how to perform the update using a non-secure session in synchronous mode. There are also APIs available to initiate an SSL session to the LDAP server. The APIs are documented in the order they have to be used. Figure 19-2 shows an overview of the APIs and the order in which they are used.
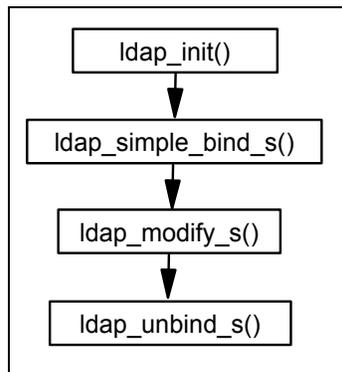
Figure 19-2   Overview of API used for updating a directory entry

In the example shown in Figure 19-2, the application will perform an update of the existing entry with the DN of uid=mjordan,ou=People,o=ibm,c=us. Example 19-2 shows the current attributes and Example 19-3 on page 613 shows the new attributes after the update has been performed.

Example 19-2   Current attributes before being updated

```
Entry: uid=mjordan,ou=People,o=ibm,c=us
Attribute: cn Value: Michael Jordan
Attribute: uid Value: mjordan
Attribute: sn Value: Jordan
Attribute: givenName Value: Michael
Attribute: telephoneNumber Value: 202-555-1234
Value: 919-555-9876
Attribute: mail Value: mjordan@yahoo.com
```

```
Attribute: objectclass Value: top
Value: person
Value: organizationalperson
Value: inetorgperson
```

Example 19-3   Attribute values after being updated

```
Entry: uid=mjordan,ou=People,o=ibm,c=us
Attribute: cn Value: Michael Jordan
Attribute: uid Value: mjordan
Attribute: sn Value: Jordan
Attribute: givenName Value: Michael
Value: Mike
Attribute: employeeNumberValue: 23
Attribute: mail Value: mjordan@yahoo.com
Attribute: objectclass Value: top
Value: person
Value: organizationalperson
Value: inetorgperson
```

As shown in Example 19-3, the common name (cn), last name (sn), userid (uid),
and e-mail address (mail) attributes remain unchanged. The first name
(givenName) was changed, the telephone numbers have been deleted, and the
employee number (employeeNumber) was added. The following descriptions
contain an overview of each API that is involved in the update process.

19.5.1 ldap_init()

The `ldap_init()` API initializes a session with an LDAP server. The server is not
actually contacted until an operation is performed that requires it, allowing
various options to be set after initialization, but before actually contacting the
host. It allocates an LDAP structure that is used to identify the connection and
maintain per-connection information. The input parameters required are the host
and port number of the LDAP server. The `ldap_init()` function returns a pointer
to an LDAP structure, which should be passed to subsequent calls to other LDAP
functions such as `ldap_simple_bind_s()` and `ldap_search_s()`.

19.5.2 ldap_simple_bind_s()

The `ldap_simple_bind_s()` function is used to authenticate a distinguished name
(DN) to a directory server. There are other APIs available to authenticate users
with a different authentication method. With LDAP Version 3 the bind API can be
skipped allowing an anonymous connection to the directory server. However,
anonymous access will have limited access on the majority of LDAP servers. The
`ldap_simple_bind_s()` API requires as input parameters: The LDAP structure ld

as returned by the `ldap_init()` API, the distinguished name (DN) of the entry performing the bind, and the password. The DN used has to have the authorities to perform the intended changes. The return code of this API indicates a successful bind or another error code.

19.5.3 ldap_modify_s()

The `ldap_modify_s()` API is a synchronous API that can be used to add, replace, and delete attributes from an existing entry. It takes several input parameters. The first one is the LDAP structure as returned by the `ldap_init()` API. The second parameter is the DN of the entry to be changed. It is not the DN used for the `ldap_simple_bind_s()` API unless the authenticated DN is the one that needs to be changed. The third parameter, named mods, is more complex. It is a NULL-terminated array of modifications to be performed to the entry. Each element of the mods array is a pointer to an LDAPMod structure. In regards to the changes described in Example 19-3 on page 613, three LDAPMod structure elements are required.

Element 1 (changes the first name):

► mod_op: Set to 0x02 (LDAP_MOD_REPLACE).

► mod_type: Specifies the name of the attribute. In this case it is givenName.

► mod_vals: The mod_vals field specifies a pointer to a NULL-terminated array of values to add, modify, or delete. In this case the pointer points to an array with three elements. The first two elements contain the first name values. The third element is a null pointer.

Element 2 (adds the employee number):

► mod_op: Set to 0x00 (LDAP_MOD_ADD).

► mod_type: Specifies the name of the attribute. In this case it is employeeNumber.

► mod_vals: The mod_vals field specifies a pointer to a NULL-terminated array of values to add, modify, or delete. In this case the pointer points to an array with two elements. The first element contains the employee number and the second a null pointer.

Element 3 (removes the telephone number):

► mod_op: Set to 0x01 (LDAP_MOD_DELETE).

► mod_type: Specifies the name of the attribute. In this case telephoneNumber.

► mod_vals: The mod_vals field specifies a pointer to a NULL-terminated array of values to add, modify, or delete. Since this element is supposed to delete

the mail attribute, mod_vals is set to NULL. The pointer can also point to a specific value to be removed.

An LDAPMod element is not necessary for last name (sn) and first name (givenName) attributes, as they remain unchanged. All modifications are performed in the order in which they are listed.

The return value of the `ldap_modify_s()` API indicates whether the modification was successful or not.

19.5.4 ldap_unbind_s()

After all entries have been processed the application must unbind from the LDAP server using, as in this example, the `ldap_unbind_s()` API. The API is used to end the connection to the LDAP server and free the resources contained in the LDAP structure that was created by the `ldap_init()` API.

19.6 Sample code to update a directory entry

Important: The sample application shown in Example 19-4 was written to help developers understand the various tasks involved to use the API to search an LDAP-based directory. It was written to provide a proof of concept.

> **Important:** The sample application shown in Example 19-4 does not cover and act on all possible exceptions, nor is it fully tested under all possible circumstances. It is a working application that can be used as an example and be extended to build a complete application.

Example 19-4   Code to update a directory using the C API

```
/* c_modify.c - simple program to modify an ldap user */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <ldap.h>

/* global variables */
static char *binddn = "cn=root";
static char *bindpwd = "password";
static char *ldaphost = "serverA.ibm.com";
static int ldapport = LDAP_PORT;
```

```
static char *ldapbase = "o=ibm,c=us";
static int referrals = LDAP_OPT_ON;
static int deref = LDAP_DEREF_NEVER;
static int ldapversion = LDAP_VERSION3;

main( int argc, char **argv )
{
   int rc;          // return code
   int i = 0;   // counter
   const char *errormsg = NULL;// error msg
   LDAP *ld;     // Ldap Object
   LDAPMod **mod;// Ldap Modification Object
   char *givenName[] = {"Michael", "Mike", NULL};// array of givenName values,
NULL terminated
   char *employeeNum[] = {"23", NULL};// array of employeeNumber values, NULL
terminated

   /* open connection to server */
   if ((ld = ldap_init(ldaphost, ldapport)) == NULL)
   {
      perror("ldap_init");
      exit(1);
   }

   // perform simple bind to server
   rc = ldap_simple_bind_s( ld, binddn, bindpwd );

   if ( rc != LDAP_SUCCESS )
   {
      errormsg = ldap_err2string(ldap_get_errno(ld));
      fprintf(stderr, "ldap_bind_s: %s \n", errormsg);
      exit(rc);
   }

   printf("Connection complete\n");

   // Construct the array of LDAPMod structures representing the attributes
   mod = (LDAPMod **)malloc((4) * sizeof(LDAPMod *));

   // Allocate the memory for each of the Mod objects
   for (i = 0; i < 3; i++)
   {
      if ((mod[i] = (LDAPMod *)malloc(sizeof(LDAPMod))) == NULL)
      {
         fprintf(stderr, "Cannot allocate memory");
      }
   }

   // set up mod object with attributes we want to modify
```

```
    // replace the current value of givenName with Mike
    mod[0]->mod_op = LDAP_MOD_REPLACE;
    mod[0]->mod_type = "givenName";
    mod[0]->mod_values = givenName;

    // Add the employeenumber of 23
    mod[1]->mod_op = LDAP_MOD_ADD;
    mod[1]->mod_type = "employeeNumber";
    mod[1]->mod_values = employeeNum;

    // Delete the attribute telephoneNumber
    mod[2]->mod_op = LDAP_MOD_DELETE;
    mod[2]->mod_type = "telephoneNumber";
    mod[2]->mod_values = NULL;

    // NULL terminate the array
    mod[3] = NULL;

    // Perform the modify operation.
    rc = ldap_modify_s(ld, "uid=mjordan,ou=People,o=ibm,c=us", mod);

    if ( rc != LDAP_SUCCESS )
    {
        errormsg = ldap_err2string(ldap_get_errno(ld));
        fprintf(stderr, "ldap_modify_s: %s \n", errormsg);
        exit(rc);
    }

    printf("Modification complete\n");

    /* unbind and exit */
    ldap_unbind_s(ld);
    printf("Connection complete\n");
    exit(rc);
}
```

Developing JNDI-based applications

Java applications, whether running as a stand-alone application, a servlet, or another form, can also utilize information stored in an LDAP-accessible directory. The industry-standard Java interface for connecting and interfacing with directories is called the Java Naming and Directory Interface (JNDI). JNDI is a Java Standard Extension. With JNDI, developers can connect seamlessly to multiple naming and directory services. They can build powerful and portable directory-enabled Java applications by using this interface. In this chapter, a sample application is provided that searches a directory, and displays the results to STDOUT. Another application is provided that updates certain attributes of a directory user. This sample application demonstrates the standard Java API calls used when working with LDAP directories. For more information on the JNDI interface and how to use it, refer to:

http://java.sun.com/products/jndi/docs.html

Any Java application, whether it is a servlet, a server application, or a client application, can be directory-enabled. Developers can exploit LDAP directory information, for example, for automatically addressing payment slips, retrieving user information at a user help desk, or performing application authentication. Developers can even serialize Java objects, such as GUI elements, into an LDAP directory and dynamically load them by all Java applications. The advantage of this method is, for example, that corporate-wide GUI design

619

requirements can be deployed and changed very easily without recompiling programs or even touching the Java programs. The Java package that allows developers to directory-enable their applications is the Java Naming and Directory Interface (JNDI) developed by Sun Microsystems, Inc. There are also other Java LDAP clients available, for example the Java LDAP client from OpenLDAP (http://www.openldap.org). This client is written directly to the LDAP protocol. This chapter shows, based on a pair of sample applications, how to use the JNDI interface. However, it does not provide a complete description of the package and its included classes. For the most current information, as well as comprehensive tips for LDAP Users section, refer to the following Web page:

http://java.sun.com/products/jndi/

20.1 The JNDI

JNDI, defined by Sun Microsystems, Inc., provides naming and directory functionality to Java programs. JNDI is an API independent of any specific directory service implementation. It enables seamless access to directory objects through multiple naming facilities.

The definition prevents, by design, the appearance of any implementation-specific artifacts in the API. The unified API is designed to cover the common case. Providing this unified interface does not imply that access to unique features of a particular service, such as LDAP, is precluded; additional classes can be added to access service-unique features. JNDI can be used by a wide range of Java programs running on servers and traditional clients. JNDI can also accommodate a thin client by specifying a service provider that provides a proxy-style protocol where access to specific naming and directory services is relegated to a server. Security is dealt with by individual service providers; however, security-related problems can be returned to the client.

As discussed above, JNDI provides a generalized naming and directory service interface. For example, JNDI could be used to retrieve files from a file system. In this case, a file system acting as a naming service could return the file that is bound to a particular file name. JNDI could also be used to access an LDAP directory, performing searches and retrieving attributes. JNDI provides an API that applications use to access a naming and directory service. The naming and directory service could be provided by any of a variety of servers, such as LDAP, NDS, or a file system. JNDI provides a Service Provider Interface (SPI) that enables access to the particular underlying directory service.

JNDI provides classes that implement a naming interface for applications, such as the file system example, that only look up names and access objects bound to names. JNDI also provides a directory interface that extends the naming interface. The directory interface adds functionality to access attributes and schema.

In JNDI terminology, a name is made up of individual components called atomic names that correspond to RDNs in LDAP. A sequence of atomic names is a compound name. An LDAP DN is a compound name. Since the underlying naming and directory services can have different name syntaxes, the SPI provides an implementation of a NameParser that can break a name into its component parts. For example, LDAP RDNs are separated by commas; DNS names are separated by periods, and so on. Composite names are compound names that span different name spaces. For example, an LDAP URL can contain both a DNS and an LDAP name, as, for instance, in `ldap://serverA.ibm.com/uid=mjordan,ou=people,o=ibm,c=us`.

Names are interpreted within a context. A context can be thought of as a particular node in the Directory Information Tree (DIT). If the current context is o=ibm,c=us, then the atomic name ou=people refers to the child node in the DIT with the DN ou=people,o=ibm,c=us. The node ou=people,o=ibm,c=us is also called a subcontext of o=ibm,c=us. A name space is traversed from context to subcontext like a file system is traversed from a directory to the directory subtree.

The DirContext interface extends the Context interface by adding operations specific to a directory service such as accessing attributes and searching. An application must establish an initial directory context as a starting point from which to do searches or traverse the DIT. The initial directory context is usually the name of an LDAP server.

JNDI does provide a mechanism for using extended operations and extended responses, and provides some implementations of these, for example, the StartTLS operation. Searches use a search filter as defined in *The String Representation of LDAP Search Filters*, RFC 2254, which is available at http://www.ietf.org/rfc/rfc2254.txt?number=2254. A SearchControls object passed to the search method can be set to control search characteristics such as the scope of the search, the number of entries returned, the time limit, etc. Also, the entire schema name space can be browsed, and object and attribute schema definitions can be retrieved.

When a directory context is established, it is passed to an environment that contains preferences and controls to access the directory service. The environment specifies the SPI to use, the security level for binding to the server, and so on. The environment is a Hashtable or Properties list of (key, value) pairs. The environment settings could be coded in the application, retrieved from the system properties, or retrieved from a file. Table 20-1 lists some of the important environment properties.

Table 20-1   Environment settings and their descriptions

Environment property	Description
java.naming.factory.initial	Contains the class name of the initial context factory. The property value should be the fully qualified class name of the factory class that is being used to create an initial context.
java.naming.provider.url	LDAP URL that specifies the LDAP server.
java.naming.ldap.version	Specifies if server supports LDAP Version 2 or 3.

Environment property	Description
java.naming.referral	Specifies if referrals should be followed, ignored, or throw an exception.
java.naming.security.authentication	Authentication method used to bind to LDAP server: None, simple, or strong.
java.naming.security.principal	Distinguished name of user to authenticate.
java.naming.security.credentials	Password or other security credential.
java.naming.security.protocol	Specifies whether the connection to the LDAP server is secure (SSL).

20.2 Searching the directory

This section explains the JNDI methods required to search a directory using the JDNI interface. Performing searches is one of the most common functions JNDI is used for. Example 20-1 shows a sample Java application that performs a search on a directory and displays the results in LDIF format to STDOUT.

Example 20-1   Java application using JNDI that performs a directory search

```
import java.util.Hashtable;
import javax.naming.ldap.InitialLdapContext;
import javax.naming.*;
import javax.naming.directory.*;

public class JavaSearch
{
    public static void main(String args[])
    {
        InitialLdapContext ctx = null;
        Hashtable hashtable = null;

        // Set up default values for LDAP info
        String url = "ldap://serverA.ibm.com:389";
        String username = "cn=root";
        String password = "password";
        String base = "o=ibm,c=us";

        try
        {
            // Set up LDAP config settings
            hashtable = new Hashtable();
            hashtable.put("java.naming.ldap.version", "3");
```

```
                hashtable.put("java.naming.factory.initial",
    "com.sun.jndi.ldap.LdapCtxFactory");
                hashtable.put("java.naming.security.authentication", "Simple");
                hashtable.put("java.naming.referral", "follow");

                hashtable.put("java.naming.provider.url", url);
                hashtable.put("java.naming.security.principal", username);
                hashtable.put("java.naming.security.credentials", password);

                // Make LDAP connection
                ctx = new InitialLdapContext(hashtable, null);
                System.out.println("Connection established");

                // Set up Search Controls
                SearchControls sc = new SearchControls();
                sc.setSearchScope(SearchControls.SUBTREE_SCOPE);

                // perform search on directory
                NamingEnumeration results = ctx.search(base,
    "(objectclass=inetorgperson)", sc);

                // loop until we have gotten all entries returned by search
                while (results.hasMore())
                {
                    // get the SearchResult object
                    SearchResult sr = (SearchResult)results.next();

                    // ouptput DN of entry
                    System.out.println(sr.getName() + "," + base);

                    // get the attributes and attribute list
                    Attributes attrs = sr.getAttributes();
                    NamingEnumeration attrList = attrs.getAll();

                    // while we have attributes
                    while (attrList.hasMore())
                    {
                        Attribute attr = (Attribute)attrList.next();

                        // Get the attribute's values
                        NamingEnumeration values = attr.getAll();
                        while (values.hasMore())
                        {
                            // output the attribute name and value
                            System.out.println(attr.getID() + "=" + values.next());
                        }
                    }
                    System.out.println();
                }
```

```
            // Close the connection to LDAP
            ctx.close();
        }
        catch (Exception ex)
        {
            System.out.println("EXCEPTION = " + ex.toString());
        }
    }
}
```

Each JNDI method used in the application will be described in detail. The sample application imports the JDNI packages shown in Example 20-2.

Example 20-2   JNDI packages that are imported

```
import java.util.Hashtable;
import javax.naming.ldap.InitialLdapContext;
import javax.naming.*;
import javax.naming.directory.*;
```

20.2.1 Creating the directory context

A context can be thought of as a bind, in terms of API calls. The context specifies to which LDAP server to connect, what DN and password to use for the bind, what authentication method to use, and so forth. An instance of the javax.naming.ldap.InitialLdapContext class needs to be created. There are several constructors for this class. However, to properly initialize the context, an environment must be provided as the parameter. This can be accomplished by instantiating a Hashtable class called *hashtable*. The next step adds the following entries to the hashtable:

► Context.INITIAL_CONTEXT_FACTORY - A constant that stores the name of the environment property for specifying the initial context factory to be used. The value of the property has to be the fully qualified class name of the factory class that will create an initial context. The application will use the default Sun LDAP factory, which is shipped with JNDI.

► Context.PROVIDER_URL - This constant holds information about the LDAP server address and its port. In this case, the server address and port are provided via a variable url, which is defined at the beginning of the application.

► Context.SECURITY_AUTHENTICATION - This constant specifies what kind of authentication is being used when binding to the directory server. The possible values depend on the service provider that is used. In this case, the application used the Sun JNDI interface, which supports the authentication

mechanisms none, simple, and strong. Other service providers, such as IBM's JNDI might also support SASL or other values. The simple authentication method uses DNs and passwords in clear text for authentication.

► Context.SECURITY_PRINCIPAL - This constant defines the DN to be used for authentication. The sample application uses the directory server's administrative DN of cn=root to authenticate.

► Context.SECURITY_CREDENTIALS - This constant defines the password to be used for authentication.

The last step of creating the directory context is using the Hashtable hashtable as a parameter for the constructor InitialLdapContext constructor. The name of the context is later being used as a "handle" to the connection. The context can be used for search, get, and update operations.

20.2.2 Performing the search

After the context has been created, a search operation can be performed using the new context. The directory context provides several `search()` methods. The methods differ by the type and number of parameters they require, but they all have one thing in common, they return an enumeration of objects. The returned objects are instances of the SearchResult class.

The sample application uses the following search method:

```
public NamingEnumeration search(String name, String filter, SearchControls
cons) throws NamingException
```

The first parameter of the selected `search()` method specifies the name of the context or object to search. In the example, this parameter has the value `o=ibm,c=us`, or the top of the directory tree. This means the search will start at the top of the tree. If the directory tree had multiple subcontexts for each organization, the search could be narrowed to search for employees within a specific organization by specifying a subcontext for the search (for example - `ou=employees,ou=Tivoli,o=ibm,c=us`).

The second parameter represents the search filter. The filter specifies the search criteria. In the example, the search filter is `(objectclass=inetorgperson)`. This filter will return all inetorgperson objects (standard person object) within the directory.

The third parameter defines the search controls. Search controls are used to control the behavior of a search operation. In this example, a search controls object *sc* is created. Two search control properties are set. The first one defines the scope of the search operation. A SUBTREE_SCOPE specifies that the

search operations start at the search base defined in the first parameter of the `search()` method and searches through all subcontexts. The search could also be limited to just the context as defined in the search base. The second property defines the maximum number of search results that are returned. A value that is used in many applications is 100. This value should always be set to avoid a multitude of problems. Imagine a directory with 200,000 entries, and somebody searched for all entries that have an e-mail address. Assuming that all entries contain an e-mail address, the search would return all 200,000 entries. In this case, the client application might not be designed or have the required resources to handle all the responses. The SearchControls class documentation contains further information on all available properties that can be defined.

The search method returns, for each directory entry found, a separate instance of SearchResult in a NamingEnumeration object.

20.2.3 Processing the search results

At this point in the application, the search has found entries that match the search filter. The search results are stored in a NamingEnumeration object results. Each element in the NamingEnumeration object is an instance of the SearchResult class and contains all attributes returned by the search. The search results need to be processed in a nested approach, such as the while loop in the example.

The program checks first if the search returned a result by using the `hasMore()` method of the NamingEnumeration class. The while loop processes as long as there are more elements in the NamingEnumeration object answer. Within the while loop, the NamingEnumeration `next()` method is used to retrieve the next element and cast it to a SearchResult object *sr*. The *sr* object holds all attributes of the directory entry in the object class Attributes. The distinguished name of the object is then outputted to stdout.

Using the `getAttributes()` method of the SearchResult class, the attributes are retrieved from the search result and stored in the Attributes attrs object. Using the `getAll()` method of the Attributes object, a NamingEnumeration object attrList is returned with all of the Attribute objects available. By looping through the attrList object, each of the Attribute objects can be retrieved with the NamingEnumeration `next()` method. Finally, with the actual Attribute object attr, all of the attributes' values can be retrieved using the `getAll()` method. This returns another NamingEnumeration object called values that contains a list of the values. By looping through this values object, the attribute name and value pair can finally be outputted to stdout. If the attribute contains more than one value, than there will be a line for each value.

After all entries have been processed, the directory context *ctx* is closed and the example code is complete.

20.3 Changing a directory entry

This section explains the JNDI methods required to modify a directory entry using the JDNI interface. Performing modifications to entries is another common JNDI function. Adding, modifying, or deleting attributes or an entire entry also requires creating a context. However, there are different methods for creating or deleting entire directory entries or, in LDAP terms, subcontexts, and for adding, modifying, or deleting attributes for an individual entry. The context's `createSubcontext()` method is used to create a new entry and the `destroySubcontext()` method to remove or delete an entry. The `modifyAttributes()` method is used to add, modify, and delete attributes for a directory entry or subcontext. The sample application shows only the more complex task of modifying attributes. This section describes the important parts of the code for creating the context, getting all attributes of the entry to changed, and performing the update on the selected entry. Example 20-3 shows the sample Java application that performs a simple modification of an entry in the directory (replaces the current givenName attribute, adds an employeenumber attribute, and removes the telephoneNumber attribute).

Example 20-3   Java application using JNDI to change a directory entry

```
import java.util.Hashtable;
import javax.naming.ldap.InitialLdapContext;
import javax.naming.*;
import javax.naming.directory.*;

public class JavaModify
{
   public static void main(String args[])
   {
      InitialLdapContext ctx = null;
      Hashtable hashtable = null;

      // Set up default values for LDAP info
      String url = "ldap://serverA.ibm.com:389";
      String username = "cn=root";
      String password = "password";
      String base = "o=ibm,c=us";

      try
      {
         // Set up LDAP config settings
         hashtable = new Hashtable();
```

```
            hashtable.put("java.naming.ldap.version", "3");
            hashtable.put("java.naming.factory.initial",
"com.sun.jndi.ldap.LdapCtxFactory");
            hashtable.put("java.naming.security.authentication", "Simple");
            hashtable.put("java.naming.referral", "follow");

        hashtable.put("java.naming.provider.url", url);
            hashtable.put("java.naming.security.principal", username);
            hashtable.put("java.naming.security.credentials", password);

            // Make LDAP connection
            ctx = new InitialLdapContext(hashtable, null);
            System.out.println("Connection established");

            // Perform modifications
            ModificationItem[] mods = new ModificationItem[3];

            // replace (update) givenName attribute with 2 values
            Attribute mod0 = new BasicAttribute("givenname", "Mike");
            mod0.add("Michael");
            mods[0] = new ModificationItem(DirContext.REPLACE_ATTRIBUTE, mod0);

            // add employeeNumber attribute
            Attribute mod1 = new BasicAttribute("employeenumber", "23");
            mods[1] = new ModificationItem(DirContext.ADD_ATTRIBUTE, mod1);

            // remove telephone number attribute
            Attribute mod2 = new BasicAttribute("telephonenumber");
            mods[2] = new ModificationItem(DirContext.REMOVE_ATTRIBUTE, mod2);

            // Perform modification of user
            ctx.modifyAttributes("uid=mjordan,ou=People," + base, mods);
            System.out.println("Modification is complete");

            // Close the connection to LDAP
            ctx.close();
            System.out.println("Connection ended");
        }
        catch (Exception ex)
        {
            System.out.println("EXCEPTION = " + ex.toString());
        }
    }
}
```

20.3.1 Creating the directory context

A context can be thought of as a bind, in terms of API calls. The context specifies to which LDAP server to connect, what DN and password to use for the bind, what authentication method to use, and so forth. An instance of the javax.naming.ldap.InitialLdapContext class needs to be created. There are several constructors for this class. However, to properly initialize the context, an environment must be provided as the parameter. This can be accomplished by instantiating a Hashtable class called *hashtable*. The next step adds the following entries to the hashtable:

► Context.INITIAL_CONTEXT_FACTORY - A constant that stores the name of the environment property for specifying the initial context factory to be used. The value of the property has to be the fully qualified class name of the factory class that will create an initial context. The application will use the default Sun LDAP factory, which is shipped with JNDI.

► Context.PROVIDER_URL - This constant holds information about the LDAP server address and its port. In this case, the server address and port are provided via a variable url, which is defined at the beginning of the application.

► Context.SECURITY_AUTHENTICATION - This constant specifies what kind of authentication is being used when binding to the directory server. The possible values depend on the service provider that is used. In this case, the application used the Sun JNDI interface, which supports the authentication mechanisms none, simple, and strong. Other service providers, such as IBM's JNDI might also support SASL or other values. The simple authentication method uses DNs and passwords in clear text for authentication.

► Context.SECURITY_PRINCIPAL - This constant defines the DN to be used for authentication. The sample application uses the directory server's administrative DN of cn=root to authenticate.

► Context.SECURITY_CREDENTIALS - This constant defines the password to be used for authentication.

The last step of creating the directory context is using the Hashtable *hashtable* as a parameter for the constructor InitialLdapContext constructor. The name of the context is later being used as a "handle" to the connection. The context can be used for search, get, and update operations.

20.3.2 Performing the modification

The first thing needed to perform modifications, is an array of ModificationItem objects. One object will be needed for each attribute that needs to be added, removed, or replaced. In the sample application a ModificationItem array named

mods is created with three elements. The next step is to fill this array with the proper ModificationItem objects. A ModificationObject constructor requires two things. First, it requires the modification to perform (ADD, REPLACE, and REMOVE). Second, it requires an Attribute object to use for the modification.

In the sample application, the first modification is to replace the user's first name (givenName attribute) with the values "Mike" and "Michael". A BasicAttribute object named mod0 is created with the attribute name of "givenName" and the initial value of "Mike". A second value "Michael" is then added to the mod0 object using the `add()` method of the BasicAttribute class. Finally, a ModificationItem is created in the first element of the mods array with a DirContext.REPLACE_ATTRIBUTE operation and the mod0 object.

The second modification is to add the employeeNumber attribute with the value "23". Again, a BasicAttribute object named mod1 is created with the attribute name of "employeeNumber" and the value of "23". Next, a ModificationItem is created in the second element of the mods array with a DirContext.ADD_ATTRIBUTE operation and the mod1 object.

The third modification is to remove the telephoneNumber attribute. A BasicAttribute object named mod2 is created with the attribute name of "telephoneNumber" with no values. Although a value could be specified, it does not make a difference because the attribute is being removed. The final ModificationItem is created in the third element of the mods array with a DirContext.REMOVE_ATTRIBUTE operation and the mod2 object.

Now that the ModifcationItem array is complete, the update to the directory entry can be performed. The InitialDirContext `modifyAttributes()` method is used to update the directory entry. The following `modifyAttributes()` method is used in the sample application:

```
public void modifyAttributes(String name, ModificationItem[] mods) throws
NamingException
```

This first parameter is a string representation of the directory entry's DN to be changed. In the sample application this value is "uid=mjordan,ou=People,o=ibm,c=us". The next parameter is the ModificationItem array mods that was created above. Assuming the call returns successfully, than the modifications are complete. Otherwise a NamingException would be thrown. After the call completes, the *ctx* is closed and the sample application is complete.

Part 5

Appendixes

We are providing a few appendixes that provide additional information on LDAP-related topics, or additional information on topics covered in this book. Specifically, we are providing DSML Version 2 information, directory integration using IBM Tivoli Directory Integrator, moving RACF users to TDBM, and schema changes that are not allowed.

A

DSML Version 2

This appendix covers:

- ► DSML introduction
- ► IBM DSML V2 service implementation
- ► IBM DSML V2 service installation, configuration and execution.
- ► Java programming examples on DSML

By the end of this appendix, you will better understand the answers to the following issues:

- ► What is DSML?
- ► Why use DSML?
- ► What is the difference of DSML and LDAP?
- ► How does IBM support DSML V2 in IBM Directory Server?
- ► What DSML structure and operations does IBM support?
- ► How to install, configure, debug and execute IBM DSML service in IBM Directory Server?
- ► How to program functions that execute DSML operations in Java?

DSML Version 2 Introduction

This section provides an introduction to the history of DSML, the different DSML versions and their differences.

DSML

Directory Services Markup Language (DSML) is an XML for representing directory information. It is a generic import/export format for directory information. Directory information in DSML can be shared between DSML-aware applications without exposing the LDAP protocol.

XML provides an effective way to present and transfer data; Directory services allow you to share and manage data, and are thus a necessary prerequisite for conducting online business; DSML is designed to make directory service more dynamic by employing XML. DSML is an XML schema for working with directories, it is defined using a Document Content Description (DCD). Thus, DSML allows XML programmers to access LDAP-enabled directories without having to write to the LDAP interface or use proprietary directory-access APIs, and provides one consistent way to work with multiple dissimilar directories.

DSML Version 1.0

DSML V1.0 was released at the end of 1999, it only provides a meta expression of directory data model and structure, and it has a lack of support for querying and updating operations to directories. In order to do queries or updates with DSML V1.0, a DSML/LDAP tool is needed such as Active Directory Services Interfaces, or dsmltools which is a set of Java utilities for handling DSML v1.0 data, such as querying LDAP directory with query result in DSML format, import DSML data into LDAP directory (For more information, please refer to the Web site (http://www.dsmltools.org).

DSML Version 2.0

DSML v2.0 was released in 2001, it provides a method for expressing directory queries, updates, and results of these operations in XML format. This version becomes more useful for most programmers.

DSML Version 2 URN
The base URN for DSML Version 2 is:

```
urn:oasis:names:tc:DSML:2:0:core
```

This URN provides the core namespace consisting of the individual operations and responses, a request envelope, a response envelope and an envelope grouping the entries, references and result of a search operation.

See Example A-1 on how to use the DSMLv2 URN.

Example: A-1   Using the DSML Version 2 URN

```
<xml:batchRequest xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core">
```

DSML v2 is not required to be a strict superset of DSML v1, but it is desirable for DSML v2 to follow the design of DSML v1 where possible.

Difference between DSML v1 and DSML v2

DSML v1 represents LDAP directories in XML, represents the 'state' of a directory.

DSML v2 represents the 'operation' that an LDAP directory can perform and the result of such operations.

Difference between DSML v2 and LDAP

The following represents the differences between DSML Version 2 and LDAP:

► Authentication: LDAP request contains authentication, DSML request is not used to authenticate the requestor. This is because that a DSML v2 document can be transported via a variety of mechanisms. But it does not mean that DSML v2 cannot be used to authenticate the requestor, in fact, DSML v2 includes an Auth request that MAY be used to associate a security principal with a collection of DSML v2 operations.

► Grouping operations: LDAP does not include a method of grouping operations to be expressed in a single request. DSMLv2 can group multiple LDAP operations to be expressed in one request document. DSML v2 specifies a simple positional correspondence between individual requests within a request document and individual responses within a response document.

► DSMLv2 eliminates a redundant level of nested element, the LDAPMessage, that is caused by the systematic translation of RFC 2251.

► Defaulting: DSMLv2 uses defaulting in a few places where LDAP does not, this is because defaulting works more naturally in XML documents than in ASN.1 structures. In DSMLv2 the string-valued elements matchedDN and errorMessage (from LDAPResult in LDAP) and attributes (from SearchRequest in LDAP) are optional and the default values are empty

string. The sizeLimit, timeLimit, and typesOnly elements (from
SearchRequest in LDAP) have default value as 0, 0, and FALSE respectively.

Typical DSML Transaction

A typical DSML transaction contains the following steps:

1. XML application sends DSML query across HTTP network.

2. DSML Service receives the query and translates into LDAP query.

3. DSML Service retrieves data from directories, and translate back into DSML format.

4. DSML Service sends the query result back to the XML application cross the HTTP network.

See Figure A-1 for a representation of these steps.
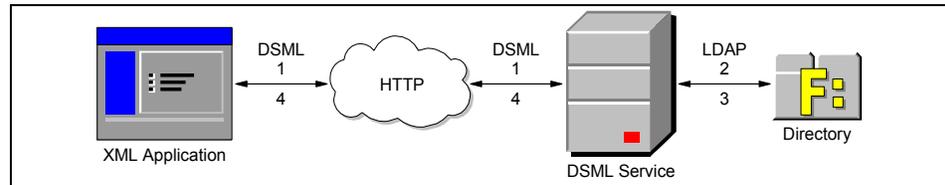
Figure A-1   Steps for a typical DSML transaction

DSML Version 2 - IBM implementation

This section discusses IBM's Implementation of DSML Version 2.

ITDS DSML Version 2 support

IBM Directory Server DSML v2 support extends the reach of the directory to Web services. Expose the directory and deliver it to Web services through XML coding. An enterprise's customers could, for example, make changes to directory data such as phone numbers or street addresses themselves over the Internet rather than calling in to customer service.

ITDS DSML Version 2 support includes the following implementation:

▶ IBM DSML Server: It provides DSML service to receive DSML request from users, executes DSML operations in LDAP server and sends DSML response back to the users.

▶ IBM DSML Client: It is used by users to submit DSML requests.

- IBM DSML structure, LDAP schema definition in DSML, request and response association, and supported DSML operations.
- IBM DSML bindings.
- IBM DSML communication between the two major IBM directory products: ITDI and ITDS.

IBM DSML Server

IBM's DSML Server provides two basic components: SOAP binding component and file binding component. A binding defines how the DSML v2 XML fragments are sent as request and responses in the context of a specific transport such as SOAP, SMTP, or a simple data file.

- The SOAP binding component allows user to submit a SOAP request over HTTP protocol. It must be deployed within an Apache SOAP v2.3 webapp. The DSML v2 'server' is a servlet in Servlet/JSP engine in the application server. See Figure A-2 for the SOAP binding component flow.
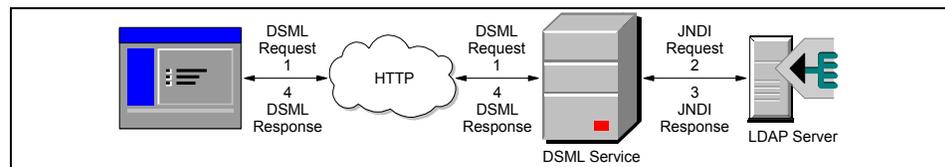
Figure A-2   SOAP binding component flow

Note that it is also feasible to convert DSML to HTTP using XSLT, and to inject it into the Web-based application flow.

- File binding component: allows a user to submit a request via XML input file. It resides on the same computer as the client, and is invoked by the DsmlFileClient. The DSML v2 'server' is a command-line program, as is typical for LDIF. The client invokes the 'server' program runs on the same computer as the server, and the input and output of the server are files consisting of DSML v2 documents. The DSML v2 server uses LDAP to communicate with the LDAP server. See Figure A-3 on page 640 for the file binding component flow.
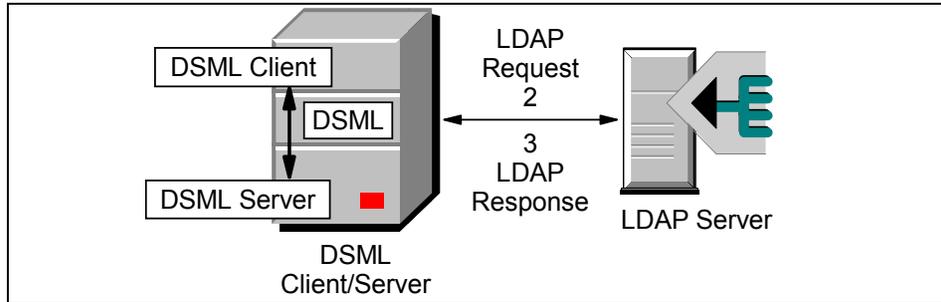
Figure A-3   File binding component flow

IBM's DSML Server installation requires a servlet-supporting application server such as WebSphere Application Server, Apache SOAP v2.3. To use DSML, Java 1.3.1 is required.

DsmlSoap Client

It is used to submit an SOAP request, the SOAP request is an XML file containing a BatchRequest to an LDAP server. The SOAP server is specified by a URL provided in a command line argument at runtime.

DsmlFile Client

It is used to submit an XML file containing a BatchRequest to an LDAP server via a DSML server sitting on the same machine as the client. Both the client and server are contained in dsml.jar, therefore, this requirement should always be satisfied.

IBM DSML Version 2 top-level structure

There are two types of DSMLv2 document: The *request* document and the *response* document. In a DSMLv2-based interaction between a client and a server there is a pairing of requests and responses: For each request document submitted by the client there is one response document produced by the server.

The top-level elements of a request fragment is a BatchRequest which contains zero, one, or many individual request elements, and the top-level elements of a response fragment is a BatchResponse which consists of zero, one or many individual response elements.

Such a batch request-response pair can be used to verify that a server is capable of processing DSMLv2 documents.

Defining directory schema in DSML

DSML also can define directory schemas, and store schema information for both object classes and attribute types. This is very useful when you want to create any unavailable schemas.

DSML object classes

See Example A-2 for the DSML schema definition for person object class.

Example: A-2   DSML schema definition for person object class

```
<dsml:dsml xmlns:dsml="http://www/dsml.org/dsml">
   <dsml:directory-schema>
      <dsml:class id="person" superior="#top" type="structural">
         <dsml:name>person</dsml:name>
         <dsml:description>Person as defined in RFC2256</dsml:description>
         <dsml:object-identifier>2.5.6.6</dsml:object-identifier>
         <dsml:attribute ref="#sn" required="true"/>
         <dsml:attribute ref="#cn" required="true"/>
         <dsml:attribute ref="#userPassword" required="false"/>
         <dsml:attribute ref="#telephoneNumber" required="false"/>
         <dsml:attribute ref="#seeAlso" required="false"/>
         <dsml:attribute ref="#description" required="false"/>
      </dsml:class>
      ... ...
   </dsml:directory-schema>
</dsml:dsml>
```

This DSML schema definition of person object class is equivalent to the 'person' object class definition in RFC2256:

```
(2.2.5.6 NAME 'person' SUP top STRUCTURAL MUST (sn $ cn ) MAY (userPassword
$ telephoneNumber $ seeAlso $ description))
```

DSML attribute types

See Example A-3 for the DSML schema definition for telephoneNumber attribute.

Example: A-3   DSML schema definition for telephoneNumber attribute

```
<dsml:directory-schema>
   <dsml:attribute-type id="telephoneNumber">
      <dsml:name>telephoneNumber</dsml:name>
      <dsml:description>telephone Number from RFC2256</dsml:description>
      <dsml:object-identifier>2.5.4.20</dsml:object-identifier>
      <dsml:syntax bound="32">1.3.6.1.4.1.1466.115.121.1.50</dsml:syntax>
      <dsml:equality>telephoneNumberMatch</dsml:equality>
      <dsml:substring>telephoneNumberSubstringMatch</dsml:substring>
   </dsml:attribute-type>
```

```
</dsml:directory-schema>
```

This DSML schema definition of telephoneNumber attribute is equivalent to the
'telephoneNumber' attribute definition in RFC2256:

```
(2.5.4.20 NAME 'telephoneNumber' EQUALITY telephoneNumberMatch SUNSTR
telephoneNumberSubstringMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.50(32))
```

Request and response association

The client and server associate an individual response in a BatchResponse with
the corresponding individual request in a BatchRequest using one (or both) of
the following methods: positional correspondence or RequestID.

In a positional correspondence, the nth response element corresponds to the nth
request element.

See Example A-4 for a valid batch request-response pair using positional
correspondence.

Example: A-4   Valid batch request-response pair using positional correspondence

```
DSMLv2 Request Document:

   <batchRequest xmlns="urn:oasis:names:tc:DSML:2:0:core">
   <modifyRequest>...</modifyRequest>
   <addRequest>...</addRequest>

   <delRequest>...</delRequest>
   <addRequest>...</addRequest>

   </batchRequest>

DSMLv2 Response Document:

   <batchResponse xmlns="urn:oasis:names:tc:DSML:2:0:core">

   <modifyResponse>...</modifyResponse>
   <addResponse>...</addResponse>

   <delResponse>...</delResponse>
   <addResponse>...</addResponse>

   <addResponse>...</addResponse>

   </batchResponse>
```

The alternative to positional correspondence is the use of the optional requestID attribute. When the client specifies a value for requestID in a request (for example, in an addRequest), the server MUST return the same value in the corresponding response (for example, in an addResponse). The client need not specify a requestID when positional correspondence is also used, although in some cases it may find this useful. For example, when using the file binding for a large file, some clients may find it more convenient to associate failed responses with requests using requestID rather than position.

A client *must not* send a request with requestID="0", as this value is reserved for unsolicited notifications.

A BatchRequest element may contain the optional XML-attribute responseOrder, which influences how the server orders individual responses within the BatchResponse. The valid values are ordered and unordered. If this attribute is omitted, the default value is ordered.

In a BatchRequest with responseOrder="ordered", the server MUST return a BatchResponse in which the individual responses maintain a positional correspondence with the individual requests.

Syntax errors

If the server detects the syntax error from the request document before performing any directory operations on behalf of the client, the response will look like Example A-5.

Example: A-5   Response to a syntax error in request document

```
<batchResponse xmlns="urn:oasis:names:tc:DSML:2:0:core">
    <errorResponse type="malformedRequest">
        <message>Unknown element 'bogusRequest'  line 87 column 4</message>
    </errorResponse>
</batchResponse>
```

The errorResponse element contains details about the error.

If the server performs one or more directory operations on behalf of the client before detecting the syntax error, the server's response contains the response element for each operation that it performed, followed by an errorResponse element. See Example A-6.

Example: A-6   Response to a syntax error after performing directory operations

```
DSML v2 request containing the syntax error:
    <batchRequest xmlns="urn:oasis:names:tc:DSML:2:0:core">
        <modifyRequest>...</modifyRequest>
        <addRequest>...</addRequest>
```

```
            <bogusRequest>...</bogusRequest>
            <addRequest>...</addRequest>s
            ...
        </batchRequest>

DSML v2 reponse to syntax error in request:
        <batchResponse xmlns="urn:oasis:names:tc:DSML:2:0:core">
            <modifyResponse>...</modifyResponse>
            <addResponse>...</addResponse>
            <errorResponse type="malformedRequest">
                <message>Unknown element 'bogusRequest'  line 87 column 4</message>
            </errorResponse>
        </batchResponse>
```

Failures

A client may produce a request document that is syntactically correct but that contains a request that fails when the provider executes it. Failure is defined as follows:

► The DSMLv2 provider was unable to connect to a server (represented as an errorResponse with type="couldNotConnect").

► The DSMLv2 provider connected to a server, but the server closed the connection without responding to the request (represented as an errorResponse with type="connectionClosed").

► The server returned an LDAPResultCode other than 0 ("success"), 6 ("compareTrue"), 5 ("compareFalse"), or 10 ("referral").

When a request execution fails, the server does not attempt to execute later requests within the document. The server produces a response element for each request element that was attempted, including the one that failed.

See Example A-7 for a DSMLv2 request that contains a request that fails.

Example: A-7   DSML v2 request that contains a request that fails

```
DSMLv2 Request containing a request that fails:
        <batchRequest xmlns="urn:oasis:names:tc:DSML:2:0:core">
            <modifyRequest>...</modifyRequest>
            <addRequest>...</addRequest>
            <delRequest>...</delRequest>
            <addRequest>...</addRequest>
        </batchRequest>

DSMLv2 Response - One request not attempted:
        <batchResponse xmlns="urn:oasis:names:tc:DSML:2:0:core">
            <modifyResponse>...</modifyResponse>
            <addResponse>...</addResponse>
```

```
        <errorResponse type="connectionClosed"/>
    </batchResponse>
```

Parallel processing

A BatchRequest element MAY contain the optional XML-attribute processing, which influences how the server can process the request elements. The valid values are sequential and parallel. If this attribute is omitted, the default value is sequential.

See Example A-8 for BatchRequest that uses parallel processing.

Example: A-8   batchRequest definition using the parallel processing attribute

```
<batchRequest xmlns="urn:oasis:names:tc:DSML:2:0:core" processing="parallel">
    ...
</batchRequest>
```

In a BatchRequest with processing="sequential", the server *must* preserve sequential semantics, that is, it behaves as already described regardless of the value of the responseOrder attribute. The effect of processing the BatchRequest *must* be as if the request elements were executed in the order they occur within the envelope.

In a BatchRequest with processing="parallel", the server MAY execute the request elements in any order. This form of processing is useful when a request contains multiple updates and the client knows that the updates are independent, as might be the case when DSMLv2 is used to bulk-load a directory. It is also useful when a request contains multiple queries and no updates.

In a BatchRequest with processing="parallel" and responseOrder="unordered", the client MUST specify a unique requestID for each individual request in the envelope. In this case, the server MAY return the responses in any order within the BatchResponse envelope; for example, in the order in which the operations complete, to improve server efficiency. If the client fails to specify a requestID for each request, the server MUST return an errorResponse with type="malformedRequest".

Resuming on error

A BatchRequest element MAY contain the optional XML-attribute onError, which influences how the server responds to failures while processing request elements. The valid values are: exit and resume. If this attribute is omitted, the default value is exit.

See for a batchRequest definition that contains the onError attribute.

Example: A-9   batchRequest definition using the onError attribute

```
<batchRequest xmlns="urn:oasis:names:tc:DSML:2:0:core" onError="resume">
   ...
</batchRequest>
```

In a BatchRequest with onError="exit", the server stops executing request elements as soon as one request element fails, and the response that is sent implicitly includes a notAttempted response for all requests that do not otherwise have a response.

If processing="parallel" and onError="exit", the server stops initiating execution of new request elements as soon as one request element fails.

If the provider does not attempt to execute a request element, but needs to provide a response in order to maintain positional correspondence, it generates an errorResponse with type="notAttempted", as shown in Example A-10.

Example: A-10   errorResponse with type="notAttempted"

```
DSMLv2 Request with parallel execution containing a request that fails:
   <batchRequest xmlns="urn:oasis:names:tc:DSML:2:0:core"
                 processing="parallel" onError="resume">
      <modifyRequest>...</modifyRequest>
      <addRequest>...</addRequest>
      <delRequest>...</delRequest>
      <addRequest>...</addRequest>
   </batchRequest>

DSMLv2 Response - two requests not successful
   <batchResponse xmlns="urn:oasis:names:tc:DSML:2:0:core">
      <modifyResponse>...</modifyResponse>
      <errorResponse type="notAttempted"/>
      <delResponse>
         <resultCode code="32" descr="noSuchObject"/>
      </delResponse>
   </batchResponse>
```

In a BatchRequest with onError="resume", the server executes the remaining request elements even though one or more requests have failed. This form of processing is most useful when processing="parallel".

IBM DSML LDAP Operations

This section discusses IBM DSML LDAP Operations.

ITDS DSML Request Structure

With the exception of extendedRequest, each individual request element contains:

► A dn attribute (as in DSMLv1) containing a distinguished name.
► Zero or more control elements representing LDAP Controls.

See Example A-11 for a few examples of LDAP request elements.

Example: A-11   LDAP request elements

```
<batchRequest xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns="urn:oasis:names:tc:DSML:2:0:core">
   <modifyRequest dn="CN=Joe Smith, OU=Dev, DC=Example, DC=Com">
      ...
   </modifyRequest>
   <addRequest dn="OU=Sales,DC=Example, DC=Com">
      ...
   </addRequest>
   <delRequest dn="CN=Alice,OU=HR,DC=Example,DC=Com">
      <control>...</control>
      <control>...</control>
   </delRequest>
   <searchRequest>
      <control>...</control>
      ...
   </searchRequest>
</batchRequest>
```

See Example A-12 for an example of an LDAP Control.

Example: A-12   LDAP Control

```
<control type="1.2.840.113556.1.4.619" criticality="true">
   <controlValue
      xsi:type="xsd:base64Binary">RFNNTHYyLjAgcm9ja3MhIQ==
   </controlValue>
</control>
```

See Example A-13 for a few examples of LDAP response elements.

Example: A-13   LDAP response elements

```
<batchResponse xmlns="urn:oasis:names:tc:DSML:2:0:core">

   ...

   <modifyResponse>
```

```
        <resultCode code="53" descr="unwillingToPerform"/>
        <errorMessage>System Attribute may not be modified</errorMessage>
    </modifyResponse>

    <addResponse>
        <resultCode code="0" descr="success"/>
    </addResponse>

    <addResponse>
        <control>...</control>
        <control>...</control>
        <resultCode code="0" descr="success"/>
    </addResponse>

    ...

</batchResponse>
```

The matchedDN and errorMessage elements are optional and default to the empty string.

The resultCode element has an optional descr attribute.

DsmlValues

The definition of DsmlValue permits the following types: UTF-8, base64Binary, and any URI. The URI type is used to indicate that the contents of the value are to be found at a location defined by the URI.

Auth

The authRequest provides a means for a client to indicate that access control for the following requests is to be interpreted as though the requests are performed by the security principal identified by the principal attribute. The value of the principal attribute is an authzId, as defined by [RFC 2829]. This can be useful if the DSMLv2 server (or an LDAP server to which the DSMLv2 server connects) is capable of supporting proxy authorization [ID-ProxyAuth].

At most one authRequest *may* occur within a BatchRequest and if it does occur, it *must* be the first request. If authRequest operations are not supported by the server to which the BatchRequest is sent, then the server *must not* process the following requests and *must* return a BatchResponse with an authResponse containing an LDAPResultCode of 'authMethodNotSupported'. If authRequest operations are supported, then if there are access rights errors, processing proceeds as for a BatchRequest without an authRequest; that is, an appropriate errorResponse is generated, etc.

See Example A-14 on page 649 for an example of an authRequest.

Example: A-14   authRequest example

```
<authRequest principal="dn:CN=Bob Rush,OU=Dev,DC=Example,DC=COM"/>
```

See Example A-15 for an example of an authResponse.

Example: A-15   authReqponse example

```
<authResponse>
    <resultCode code="0"/>
</authResponse>
```

Modify

DSMLv2 specifies each attribute modification by attaching an operation attribute to an attr element. As in LDAP, an operation can be add, delete, or replace.

See Example A-16 for an example of a modifyRequest.

Example: A-16   modifyRequest example

```
<?xml version="1.0"  encoding="UTF-8"?>
<dsml:batchRequest xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <dsml:modifyRequest  dn="cn=chunhui yang,o=ibm">
      <modification name="telephoneNumber" operation="add">
         <value>123 456 7890</value>
         <value>919 824 9855</value>
      </modification>
      <modification name="sn" operation="replace">
         <value>Richard</value>
      </modification>
   </dsml:modifyRequest>
</dsml:batchRequest>
```

See Example A-17 for an example of a modifyResponse.

Example: A-17   modifyResponse example

```
<modifyResponse>
   <resultCode code="53" descr="unwillingToPerform"/>
   <errorMessage>System Attribute may not be modified</errorMessage>
</modifyResponse>
```

Search

The DSMLv2 search encoding is based on the LDAP search encoding, but with some changes as described in Section A. In the searchRequest encoding:

▶ *baseObject*. Following DSMLv1 conventions, the distinguished name of the search base is expressed as the XML attribute dn.

```
<searchRequest dn="OU=Marketing,DC=Example,DC=COM" />
```

▶ *sizeLimit, timeLimit, typesOnly*. These elements default to 0, 0, and FALSE respectively.

▶ *attributes*. In RFC 2251, attributes is a sequence of attribute names, which is translated into a sequence of elements containing attribute names.

See Example A-18 for an example of the attributes element.

Example: A-18   attributes element example

```
<attributes>
   <attribute name="sn"/ >
   <attribute name="givenName"/>
   <attribute name="title"/>
</attributes>
```

See Example A-19 for a full SearchRequest example.

Example: A-19   searchRequest example

```
<?xml version="1.0"  encoding="UTF-8"?>
<dsml:batchRequest xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <dsml:searchRequest
       dn="o=ibm"
       scope="wholeSubtree"
       derefAliases="neverDerefAliases"
       sizeLimit="1000">
   <dsml:filter>
       <dsml:and>
           <dsml:substrings name="givenName">
               <initial>S</initial>
           </dsml:substrings>
           <dsml:equalityMatch name="objectclass">
               <value>inetorgperson</value>
           </dsml:equalityMatch>
       </dsml:and>
   </dsml:filter>
   <dsml:attributes>
       <dsml:attribute name="cn"/>
       <dsml:attribute name="sn"/>
```

```
        <dsml:attribute name="l"/>
    </dsml:attributes>
    </dsml:searchRequest>
</dsml:batchRequest>
```

The response to a searchRequest is logically called a searchResponse.
According to RFC 2251, a search response contains:

► Zero to many searchResultEntry
► Zero to many searchResultReference
► One searchResultDone

DSMLv2 permits wrapping all of these related elements into one
searchResponse envelope.

Each searchResultEntry, searchResultReference, and searchResultDone MAY
have zero or more LDAP controls, consistent with RFC 2251.

See Example A-20 for an example of a searchResultEntry (with terminating
searchResultDone).

Example: A-20   searchResultntry example

```
<searchResponse>
    <searchResultEntry dn="OU=Development,DC=Example,DC=COM">
        <attr name="allowedAttributeEffective">
            <value>description</value>
            <value>ntSecurityDescriptor</value>
            <value>wwwHomepage</value>
        </attr>
    </searchResultEntry>
    <searchResultEntry dn="CN=David,OU=HR,DC=Example,DC=COM">
        <attr name="objectclass"><value>person</value></attr>
        <attr name="sn"><value>Johnson</value></attr>
        <attr name="givenName"><value>David</value></attr>
        <attr name="title"><value>Program Manager</value></attr>
    </searchResultEntry>
    <searchResultEntry dn="CN=JSmith, OU=Finance,DC=Example,DC=COM">
        <attr name="objectclass"><value>top</value></attr>
        <attr name="objectclass"><value>person</value></attr>
        <attr name="objectclass"><value>organizationalPerson</value></attr>
        <attr name="sn"><value>Smith</value></attr>
    </searchResultEntry>
    <searchResultDone>
        <control type="1.2.840.113556.1.4.621" criticality="false">
            <controlValue xsi:type="xsd:base64Binary">
                U2VhcmNoIFJlcXVlc3QgRXhhbXBsZQ==
            </controlValue>
        </control>
```

```
            <resultCode code="0"/>
        </searchResultDone>
</searchResponse>
```

See Example A-21 for an example of a searchResultReference.

Example: A-21   searchResultReference example

```
<searchResponse>
    <searchResultReference>
        <ref>ldap://srv01.example.com/OU=Marketing,DC=Example,DC=COM</ref>
        <ref>ldap://srv05.fabrikam.com/DC=Fabrikam,DC=COM</ref>
    </searchResultReference>
    ...
</searchResponse>
```

Add

See Example A-22 for an example of an addRequest.

Example: A-22   addRequest example

```
<?xml version="1.0"  encoding="UTF-8"?>
<dsml:batchRequest xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
                    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <dsml:addRequest  dn="cn=chunhui yang,o=ibm">
        <attr name="objectclass">
            <value>top</value>
        </attr>
        <attr name="objectclass">
            <value>person</value>
        </attr>
        <attr name="objectclass">
            <value>organizationalPerson</value>
        </attr>
        <attr name="objectclass">
            <value>inetorgperson</value>
        </attr>
        <attr name="sn">
            <value>Yang</value>
        </attr>
        <attr name="givenName">
            <value>chunhui</value>
        </attr>
        <attr name="title">
            <value>ITDS consultant</value>
        </attr>
    </dsml:addRequest>
```

```
</dsml:batchRequest>
```

See Example A-23 for an example of an addResponse.

Example: A-23   addResponse example

```
<addResponse>
    <resultCode code="0"/>
    <errorMessage>completed</errorMessage>
</addResponse>
```

Delete
See Example A-24 for an example of a delRequest.

Example: A-24   delRequest example

```
<?xml version="1.0"  encoding="UTF-8"?>
<dsml:batchRequest xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <dsml:delRequest  dn="cn=deleteperson,o=ibm">
   </dsml:delRequest>
</dsml:batchRequest>
```

See Example A-25 for an example of a delResponse.

Example: A-25   delResponse example

```
<delResponse matchedDN="OU=HR,DC=Example,DC=COM">
   <resultCode code="32" descr="noSuchObject"/>
   <errorMessage>DSDEL::230234</errorMessage>
</delResponse>
```

ModifyDN
See Example A-26 for an example of a modDNRequest.

Example: A-26   modDNRequest example

```
<modDNRequest dn="CN=Alice Johnson,DC=Example,DC=COM"
   newrdn="Alice Weiss"
   deleteoldrdn="true"
   newSuperior="OU=Marketing,DC=Example,DC=COM"/>
```

See Example A-27 on page 654 for an example of a modDNResponse.

Example: A-27   modDNResponse example

```
<modDNResponse>
<resultCode code="0" descr="success"/>
</modDNResponse>
```

Compare
See Example A-28 for an example of a compareRequest.

Example: A-28   compareRequest example

```
<compareRequest dn="CN=Johnson,OU=HR, DC=Example,DC=COM">
    <assertion name="sn"><value>Johnson</value></assertion>
</compareRequest>
```

See Example A-29 for an example of a compareResponse.

Example: A-29   compareResponse example

```
<compareResponse>
    <resultCode code="6" descr="compareTrue"/>
</compareResponse>
```

Extended Operation
See Example A-30 for an example of an extendedRequest.

Example: A-30   extendedRequest example

```
<extendedRequest>
    <requestName>1.3.563.52.424</requestName>
    <requestValue
        xsi:type="xsd:base64Binary">TFNNTHYyLjAgcm9ja3MhIQ==
    </requestValue>
</extendedRequest>
```

See Example A-31 for an example of an extendedResponse.

Example: A-31   extendedResponse example

```
<extendedResponse>
    <resultCode code="0"/>
    <response xsi:type="xsd:base64Binary">RFNNTHYyLjAgcm9ja3MhIQ==</response>
</extendedResponse>
```

Bindings

DSMLv2 defines two normative bindings:

► A SOAP request/response binding
► A file binding that serves as the DSMLv2 analog of LDIF

SOAP Binding

The following describes the DSMLv2 SOAP [W3C SOAP] request/response binding.

The namespace for DSMLv2 is "urn:oasis:names:tc:DSML:2:0:core". This namespace is used at the top-level element of the <body> of each SOAP request and response. Default namespace designations *may* be used.

All SOAP requests and responses in this binding MUST use the xml encoding "UTF-8".

Each SOAP request body contains a single batchRequest. A SOAP node SHOULD indicate in the 'SOAPAction' header field the element name of the top-level element in the <body> of the SOAP request.

Each SOAP response body contains a single batchResponse.

A SOAP Fault is used only when an error occurs outside the scope of DSMLv2 processing. For example, the SOAP Server is not able to find or connect to a DSMLv2 server to process a DSMLv2 document. If errors happen during DSMLv2 processing, then they are conveyed as a DSMLv2 response document in the SOAP response message.

See Example A-32 for an example of a SOAP request.

Example: A-32   SOAP request example

```
<se:Envelope xmlns:se="http://schemas.xmlsoap.org/soap/envelope/"
   se:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <se:Body xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core">
      <dsml:batchRequest>
         <dsml:modifyRequest>...</dsml:modifyRequest>
         <dsml:addRequest>...</dsml:addRequest>
         ...
      </dsml:batchRequest>
   </se:Body>
</se:Envelope>
```

See Example A-33 on page 656 for an example of a SOAP response.

Example: A-33   SOAP response example

```
<se:Envelope xmlns:se="http://schemas.xmlsoap.org/soap/envelope/"
    se:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <se:Body xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core">
        <dsml:batchResponse>
            <dsml:modifyResponse>...</dsml:modifyResponse>
            <dsml:addResponse>...</dsml:modifyResponse>

            ...
        </dsml:batchResponse>
    </se:Body>
</se:Envelope>
```

See for an example of a SOAP fault.

Example: A-34   SOAP Fault example

```
<se:Envelope xmlns:se="http://schemas.xmlsoap.org/soap/envelope/"
se:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <se:Body>
        <se:Fault>
            <faultcode>se:Server</faultcode>
            <faultstring>Server Error</faultstring>
            <detail>Cannot connect to a DSMLv2 server</detail>
        </se:Fault>
    </se:Body>
</se:Envelope>
```

This binding does not specify any SOAP headers.

A minimal implementation supports DsmlValue URIs of type file, which are evaluated by the client provider using the security context associated with the client. Individual implementations *may* support additional URI types. If the client provider is unable to resolve the URI to a value that can transferred to the server, then the provider *must* return an errorResponse with type="unresolvableURI".

Authentication in this binding is the ID and password which comes in as part of HTTP authentication, and is reused to bind to LDAP.

File binding

The file binding is an alternative to the LDAP Data Interchange Format (LDIF) described by [RFC 2849]. Its primary advantages over LDIF are:

► Use of XML, which is more natural for many clients to generate and to parse than LDIF. Also benefits from the comparative wealth of tools.

► Formalization of output on error conditions, such as in the event the directory server is unavailable or the directory server returns an LDAP error.

The top-level document for the input file is an element of type BatchRequest with name batchRequest. The top-level document for the output file is an element of type BatchResponse with name batchResponse.

A minimal implementation supports DsmlValue URIs of type file, which are evaluated by the command-line program using the security context associated with the process running the command-line program. Individual implementations MAY support additional URI types. If the client provider is unable to resolve the URI to a value that can transferred to the server, then the command-line program MUST return an errorResponse with type="unresolvableURI".

The file binding authenticates to the directory using the user identity and password with which the command-line program was invoked.

DSML communication between ITDI and ITDS

ITDI is IBM's another directory product which focuses on directory integration practice. It can acts as a DSML client or a DSML service to communicate with ITDS DSML server.

ITDS DSML Client to ITDI DSML Service

ITDS DSML client application sends DSML request to ITDI, and ITDI HTTP EH will trigger an AssemblyLine which uses HTTP client connector and SOAP parser to deconstruct the DSML request, connects to ITDS server and executes the operation defined in the DSML request, and then sends the DSML response back to ITDS DSML client application.

ITDI DSML Client to ITDS DSML Server

ITDI can uses its HTTP client connector to send DSML request, ITDS DSML server receives the DSML request, parses the DSML document, convert the DSML operation into JNDI operation, executes the requested operation in the LDAP server via JNDI, and sends DSML response back to ITDI.

ITDS DSML Service Deployment

This section covers the detailed steps of ITDS DSML service installation, configuration, and execution.

Installation

In order to install ITDS DSML Service, you will need to download the DSML.zip file, install application server (WAS), install SOAP, and then install DSML into WAS. These steps are described in the following section.

DSMLzip file

Directory Services Markup Language (DSML) is installed as a .zip file named DSML.zip in the *installpath*/idstools (or *installpath*\idstools for Windows systems) directory when you install the Web Administration Tool.

> **Note:** During standard IBM Directory Server installation, you must select the WebAdmin package.

After you unzip the DSML.zip file, the DSML.zip file can be found in your <LDAPHome>\idstools directory, the following files are contained in the .zip file.

DSMLReadme.txt

Describes the files in the package and more detailed instructions on how to install and configure DSML and placement of .jar files.

dsml.pdf

Describes how to use DSML. This file is in PDF format.

dsml.htm

Describes how to use DSML, in HTML format.

> **Note:** The Web Administration Tool is NOT for the faint of heart: Requires specific Java files and CLASSPATH setup, knowledge of installing apps into the Application Server, plus knowledge of SOAP and XML.

Install Application Server (WAS)

The Application Server is required.The embedded Websphere Application Server 5.0 Express provided with the IBM Directory Server installation is preferred. Tomcat is also supported. The following steps should be performed when installation the application server:

► Install IBM Directory 5.2 with WebAdmin package and Websphere Application Server and GSKit6.

► Configure your ITDS 5.2 Admin DN and Password, and database (in order to test WAS with the Web Admin).

► Make sure the Websphere Application Server Express starts successfully.

► Type the following commands in the installation folder\appsrv\bin directory:

```
In Windows: startServer server1
In Unix: startServer.sh server1
```

► Test WAS by starting the WebAdmin at the following default URL:

```
http://<hostname>:9080/IDSWebApp/IDSjsp/Login.jsp
```

► If desired, go ahead and configure your new 5.2 LDAP server into WAS.

Install Java SDK 1.3.1

Java SDK 1.3.1 (not just the JRE) is needed. WebSphere Express contains an acceptable version in the ...\appsrv\java\ directory. The preferred level of the 1.3.1 SDK is Service Release 2 or greater. This can be obtained at:

```
http://www.alphaworks.ibm.com/aw.nsf/download/xml4j
```

On this page, download File: XML4J-bin.4.1.2.zip. The files needed from this zip file are xercesImpl.jar, xmlParserAPIs.jar.

Install SOAP

To install:

► Apache SOAP 2.3 must be installed into the Application Server before installing our DSML. This can be obtained at:

```
http://xml.apache.org/dist/soap/version-2.3
```

On this page, download file soap-bin-2.3.zip. The files needed from this zip file are soap.war, soap.jar.

► Some specific Java packages must be downloaded from the Web in order to get some of the Java .jar files required by DSML. (See Table A-1 for the Java files that are needed.)

Unzip the packages and copy the .jar files to <WAS>\appsrv\lib.

Copy the soap.war file to <WAS>\appsrv\installableApps.

► Configure Apache SOAP 2.3 into WAS.

Set your JAVA_HOME and CLASSPATH to values discussed in the DSMLreadme.txt file.

– Download the 5 JAR files listed in Table A-1 to install the DSML Server. These files are required for the install.bat or install.sh file to work.

Table A-1   JAR files needed to install the DSML Server

Filename	Description	Download location
mail.jar	Java Mail	http://java.sun.com/products/javamail/

Filename	Description	Download location
activation.jar	JavaBeans	http://java.sun.com/products/javabeans/glasglow/jaf/html
XercesImpl.jar	XML4Java	http://www.alphaworks.ibm.com/aw.nsf/download/xml4j
XMLParserAPIs.jar	XML4Java	http://www.alphaworks.ibm.com/aw.nsf/download/xml4j
soap.jar	Apache SOAP	http://xml.apache.org/dist/soap/version-2.3/

– Set the CLASSPATH variable to point all of the jar files you downloaded.

CLASSPATH should point to soap.jar, xercesImpl.jar, xmlParserAPIs.jar xerces.jar, activation.jar, and mail.jar.

Additional classpath setting that is not included in the dsmlreadme.txt file and WAS:

```
appsrv\installedApps\DefaultNode\soap.war.ear\soap.war\WEB-INF\classes
```

Such as:

```
C:\PROGRA~1\IBM\LDAP\appsrv\installedApps\DefaultNode\soap.war.ear\soap.
war\WEB-INF\classes;
```

See Example A-35 for an example of classpath settings.

Example: A-35   CLASSPATH settings

```
.;C:\PROGRA~1\IBM\LDAP\appsrv\lib\jaf-1.0.2\activation.jar;
C:\PROGRA~1\IBM\LDAP\appsrv\lib\javamail-1.3.1\mail.jar;
C:\PROGRA~1\IBM\LDAP\appsrv\lib\xml4j-4_2_2\XercesImpl.jar;
C:\PROGRA~1\IBM\LDAP\appsrv\lib\xml4j-4_2_2\XMLParserAPIs.jar;
C:\PROGRA~1\IBM\LDAP\appsrv\installedApps\DefaultNode\soap.war.ear\soap.war\WEB
-INF\classes;
```

– Set the JAVA_HOME variable to <WASinst>/java/.

JAVA_HOME must point to a true Java 1.3.1 SDK (not just a JRE). The <WAS>/appsrv/java directory contains an acceptable 1.3.1 JDK.

– Set the PATH variable to <WASinst>/java/bin.

– Make sure the file soap.war is in the <WAS>/appsrv/installableApps directory.

– GSKit6 (which comes with our product) contains several .jar files that are needed by DSML.

► Run the following WAS command (one long line).

The command shown in Example A-36is the Windows version, and you should replace '<WAS>' with the directory where the WAS 'appsrv' directory exists.

Example: A-36   WAS command

```
<WAS>\appsrv\bin\wsadmin.bat  -conntype NONE -node DefaultNode
 -c "$AdminApp install {<WAS>/appsrv/installableApps/soap.war}
 {-configroot \"<WAS>\config\" -node DefaultNode -usedefaultbindings
 -nodeployjb -appname soap.war -context \"soap\"}
```

> **Note:** Make sure all the back-slashes, double-quotes, and backslashes are correct.

See Example A-37 for an example of how to create a soapinstall.bat file to run the WAS command.

Example: A-37   soapinstall.bat file

```
wsadmin.bat -conntype NONE -c "$AdminApp install {C:\Program
Files\IBM\LDAP\appsrv/installableApps/soap.war} {-configroot \"C:\Program
Files\IBM\LDAP\config\"  -node DefaultNode -usedefaultbindings -nodeployejb
-appname soap.war -contextroot \"soap\"}"
```

After the installation, you will see something similar to Example A-38.

Example: A-38   soapinstall.bat output messages

```
C:\Program Files\IBM\LDAP\appsrv\bin>soapinstall

C:\Program Files\IBM\LDAP\appsrv\bin>wsadmin.bat -conntype NONE -c "$AdminApp
install {C:\Program Files\IBM\LDAP\appsrv/installableApps/soap.war}
{-configroot \"C:\Program Files\IBM\LDAP\config\"  -node DefaultNode
-usedefaultbindings -nodeployejb -appname soap.war -contextroot \"soap\"}"

WASX7357I: By request, this scripting client is not connected to any server
process. Certain configuration and application operations will be available in
local mode.
ADMA6016I: Add to workspace META-INF/application.xml
ADMA6017I: Saved document C:\Program
Files\IBM\LDAP\appsrv\wstemp\Scriptf95c04264e\workspace\cells\DefaultNode\appli
cations\soap.war.ear\deployments\soap.war\META-INF\ibm-applcation-bnd.xmi
ADMA6016I: Add to workspace META-INF/ ibm-applcation-bnd.xmi
ADMA6017I: Saved document C:\Program
Files\IBM\LDAP\appsrv\wstemp\Scriptf95c04264e\workspace\cells\DefaultNode\appli
cations\soap.war.ear\deployments\soap.war\META-INF\MANIFEST.MF
ADMA6016I: Add to workspace META-INF/ MANIFEST.MF
```

```
ADMA6017I: Saved document C:\Program
Files\IBM\LDAP\appsrv\wstemp\Scriptf95c04264e\workspace\cells\DefaultNode\appli
cations\soap.war.ear\deployments\soap.war\WEB-INF\web.xml
ADMA6016I: Add to workspace soap.war/WEB-INF/web.xml
ADMA6017I: Saved document C:\Program
Files\IBM\LDAP\appsrv\wstemp\Scriptf95c04264e\workspace\cells\DefaultNode\appli
cations\soap.war.ear\deployments\soap.war\WEB-INF\ibm-web-bnd.xmi
ADMA6016I: Add to workspace soap.war/WEB-INF/ibm-web-bnd.xml
ADMA5005I: Application soap.war configured in WebSphere repository
ADMA5037I: Starting backup of app at C:\Program
Files\IBM\LDAP\appsrv\wstemp\Scriptf95c04264e\workspace\cells\DefaultNode\appli
cations\soap.war.ear
ADMA5037I: Completed backup of app at C:\Program
Files\IBM\LDAP\appsrv\wstemp\Scriptf95c04264e\workspace\cells\DefaultNode\appli
cations\soap.war.ear
ADMA5037I: Application binaries saved in C:\Program
Files\IBM\LDAP\appsrv\wstemp\Scriptf95c04264e\workspace\cells\DefaultNode\appli
cations\soap.war.ear
ADMA5037I: Deleting directory tree
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\app_f95c0454e5
ADMA5011I: Cleanup of temp dir for app soap.war done.
ADMA5031I: Application soap.war installed successfully
```

► Stop and restart the WAS server after a successful SOAP install.

   You will see information similar to Example A-39 during the restart process.

Example: A-39   Restart WAS output

```
C:\Program Files\IBM\LDAP\appsrv\bin>stopServer server1
ADMU0116I: Tool information is being logged in file C:\Program
          Files\IBM\LDAP\appsrv\logs\server1\stopServer.log
ADMU3100I: Reading configuration for server: server1
ADMU3201I: Server stop request issued. Waiting for stop status.
ADMU4000I: Server server1 stop completed.

C:\Program Files\IBM\LDAP\appsrv\bin>startServer server1
ADMU0116I: Tool information is being logged in file C:\Program
          Files\IBM\LDAP\appsrv\logs\server1\startServer.log
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server1 open for e-business; process id is 900
```

Install DSML into WAS

To install:

► Make sure the WAS Express server has restarted successfully.

► Test SOAP is working through WAS by opening a browser and going to the
  URL:

```
http://<yourmachinename>:9080/soap/servlet/rpcrouter
```

► You should see a page with a message similar to `SOAP RPC Router - Sorry,`
   `we do not speak via HTTP GET ... use HTTP POST`, as shown in Figure A-4.
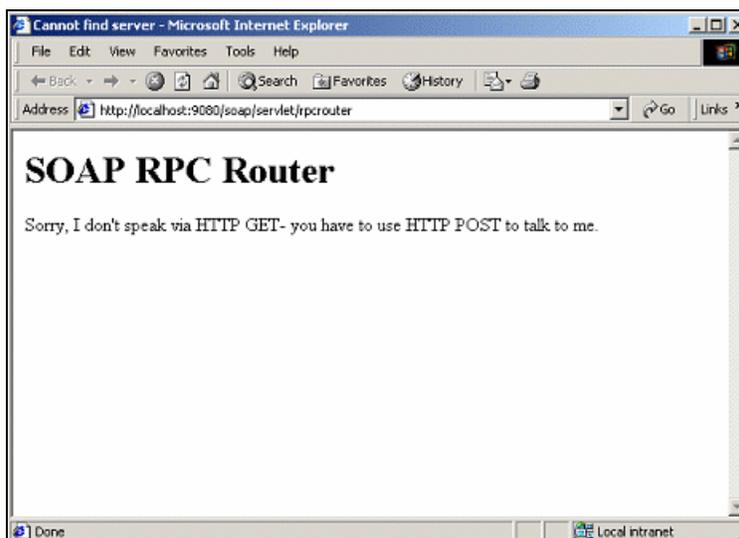
Figure A-4   Testing SOAP through WAS

► Install DSML using the following commands from the c:\dsml\ directory:

   – On Windows platforms:

      ```
      install <SOAPHomeDir> <RPCRouterURL>
      ```

   – On UNIX platforms:

      ```
      chmod u+x install.sh
      ./install.sh <SOAPHomeDir> <RPCRouterURL>
      ```

   See Example A-40 for a Windows example.

Example: A-40   Installing DSML on Windows

```
install  <WAS>\appsrv\installedApps\DefaultNode\soap.war.ear\soap.war
http://<yourmachinename>:9080/soap/servlet/rpcrouter
```

> **Note:** If there is a space (such as C:\Program Files\...) in your
> JAVA_HOME, you should quote "%JAVA_HOME%" in the install.bat file.

You should see some files copied with a message at the end that says if there
were no error messages the install was successful, as shown in
Example A-41. There should not be any Java exceptions.

Example: A-41   DSML install output

```
C:\Program Files\IBM\LDAP\idstools\DSML>install
C:\Progra~1\IBM\LDAP\appsrv\installedApps\DefaultNode\soap.war.ear\soap.war
http://lcoalhost:9080/soap/servlet/rptrouter
    1 file(s) copied.
.\jars\auibase.jar
.\jars\dsml.jar
.\jars\IBMLDAPJavaBer.jar
.\jars\regex4j.jar
    4 file(s) copied.
.\jars\auibase.jar
.\jars\dsml.jar
.\jars\IBMLDAPJavaBer.jar
.\jars\regex4j.jar
    4 file(s) copied.
    1 file(s) copied.
    1 file(s) copied.
Verified existence of logs directory:
C:\Progra~1\IBM\LDAP\appsrv\installedApps\DefaultNode\soap.war.ear\soap.war\log
s.
Verified existence of WEB-INF/lib directory:
C:\Progra~1\IBM\LDAP\appsrv\installedApps\DefaultNode\soap.war.ear\soap.war\WEB
-INF\lib.
Deploying the eCopy service....
Verify that its there
Deployed Services:
Urn:oasis:.namesLtcLDSML:2:0:core
If you have not received any errors during the install, installation is now
complete.  Please restart your application server.

C:\Program Files\IBM\LDAP\idstools\DSML>
```

If you see the Java exception as shown in Example A-42, it means that your localhost is not found.

Example: A-42   Java exception for localhost not found

```
Exception in thread "main" [SOAPException: faultCode=SOAP-ENV:Client; msg=Error
opening socket: java.net.UnknownHostException:
localhost;targetException=java.lang.IllegalArgumentException: Error opening
socket: java.net.UnknownHostException: lcoalhost]
    at org.apache.soap.transport.http.SOAPHTTPConnection.send(Unknown Source)
    at org.apache.soap.rpc.Call.invoke(Unknown Source)
    at org.apache.soap.server.ServiceManagerClient.invokeMethod(Unknown Source)
    at org.apache.soap.server.ServiceManagerClient.deploy(Unknown Source)
    at org.apache.soap.server.ServiceManagerClient.main(Unknown Source)
```

If you see the Java exception as shown in Example A-43, it means that your class path for those required jar files are not set correctly.

Example: A-43   Java exception for CLASSPATH not being set correctly

```
[Error] Thu Dec 11 19:56:38 EST 2003 <== DsmlSoapClient.main(String[])
(Exception)
java.io.FileNotFoundException:
   C:\Program Files\IBM\LDAP\idstools\DSML\DsmlRequest.xml
      (The system cannot find the file specified)
   at java.io.FileInputStream.open(Native Method)
   at java.io.FileInputStream.<init>(FileInputStream.java:103)
   at java.io.FileInputStream.<init>(FileInputStream.java:66)
   at sun.net.www.protocol.file.FileURLConnection.connect
      (FileURLConnection.java:69)
   at sun.net.www.protocol.file.FileURLConnection.getInputStream
      (FileURLConnection.java:156)
   at java.net.URL.openStream(URL.java:960)
   at org.apache.xerces.impl.XMLEntityManager.setupCurrentEntity
      (Unknown Source)
   at org.apache.xerces.impl.XMLVersionDetector.determineDocVersion
      (Unknown Source)
   at org.apache.xerces.parsers.XML11Configuration.parse(Unknown Source)
   at org.apache.xerces.parsers.DTDConfiguration.parse(Unknown Source)
   at org.apache.xerces.parsers.XMLParser.parse(Unknown Source)
   at org.apache.xerces.parsers.DOMParser.parse(Unknown Source)
   at org.apache.xerces.jaxp.DocumentBuilderImpl.parse(Unknown Source)
   at javax.xml.parsers.DocumentBuilder.parse(DocumentBuilder.java:124)
   at com.ibm.ldap.dsmlClient.DsmlSoapClient.main(DsmlSoapClient.java:205)
```

► When DSML is successfully installed, you will receive the messages "Deploying Service" and "Verifying". You are then prompted to restart your application server.

SSL with DSML

The following documentation should be reviewed to help set up SSL with DSML.

► DSMLReadme.txt file (located inside the DSML.zip file)
► IBM Directory Server 5.1 Administration Guide - Chapter 7
► IBM Directory Server 5.1 Installation and Configuration Guide - Appendix G
► GSKit documentation

Some of the main tasks associated with setting up SSL are:

► Creating the key database/trust store database with the ikeyman utility on the Directory Server machine.

► Creating a self-signed certificate for LDAP server (or acquiring a real certificate).

- Exporting the LDAP server's SSL certificate into the key database.
- Importing the LDAP certificate to WAS for 1-way handshaking.
- Creating a self-signed certificate for SOAP server if 2-way SSL handshaking is desired.
- Updating the <JAVA_HOME>\jre\lib\security\java.security file with proper provider information.
- Updating SOAP's DSMLSSLConfig.xml file to recognize the certificate.

Configuration

DSML Service configuration settings include the logging and SSL settings on the server.

Logging

The DSML server will log to the file defined by the LogFile parameter in the trace.properties file, the default LogFile name is DSMLLLog.txt. What gets logged is defined by four configurable options that are also set in the trace.properties file contained in <SOAPHomeDir>\WEB-INF.

The four options and descriptions that can be set in the trace.properties file are:

- trace.information: Possible values are true and false. When set to false, log messages that are informational in nature will be suppressed. Otherwise, they will be written to the log file.
- trace.diagnostic: Possible values are true and false. When set to false, log messages that are diagnostic (tracing) in nature will be suppressed. Otherwise, they will be written to the log file.
- trace.warning: Possible values are true and false. When set to false, warning messages will be suppressed. Otherwise, they will be written to the log file.
- trace.error: Possible values are true and false. When set to false, error messages will be suppressed. Otherwise, they will be written to the log file.

Note: As more tracing options are turned on, performance will drop.

See Example A-44 on page 666 for an example of the trace.properties file.

Example: A-44   Logging options in the trace.properties file

```
trace.information=false
trace.diagnostic=false
trace.warning=true
```

```
trace.error=true
```

SSL

Modify the java.security file in your %JAVA_HOME%\jre\lib\security directory as follows:

▶ After the line "security.provider.1=sun.security.provider.Sun" place the following 2 lines (if you have other providers installed, the numbers may be different):

```
security.provider.2=com.ibm.crypto.provider.IBMJCE
security.provider.3=com.ibm.crypto.provider.IBMJCA
```

▶ The following options must be set in the DSMLSSLConfig.xml file located in <SOAPHome>\WEB-INF\classes\security directory for SSL to be used.

  – *SSL:* Toggles the use of SSL. Once SSL is turned "on," all bindings between the DSML server and an LDAP server will be made using SSL. If the DSML server attempts to bind to a non-secure port on an LDAP server with SSL turned on, it will fail.

  – *dsmlKeyRingFile:* Specifies the fully qualified pathname to the Keyring file to be used in secure connections.

  – *dsmlKeyRingPassword:* Password for Keyring File.

  – *dsmlKeyRingFiletype:* Keyring file type.

  – *dsmlTrustStoreFile:* Specifies the fully qualified pathname to the trust store file to be used in secure connections.

  – *dsmlTrustStorePassword:* Password for trust store file.

  – *dsmlTrustStoreFiletype:* Trust store file type.

> **Note:** The DSMLSSLConfigFileSchema.xsd file must be in the %JAVA_HOME%/jre directory. (This is done by default during installation.)

See Example A-45 for an example of the DSMLSSLConfig.xml file.

Example: A-45   DSMLSSLConfig.xml file

```
<?xml version="1.0" encoding="UTF-8" ?>
<DsmlSSLConfigxmlns="sslConfig"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <SSL>off</SSL>
        <dsmlKeyRingFile>d:\keyrings\namtp2.jks</dsmlKeyRingFile>
        <dsmlKeyRingPassword>secret</dsmlKeyRingPassword>
        <dsmlKeyRingFiletype>jks</dsmlKeyRingFiletype>
        <dsmlTrustStoreFile>d:\keyrings\namtp2.jks</dsmlTrustStoreFile>
```

```
        <dsmlTrustStorePassword>secret</dsmlTrustStorePassword>
        <dsmlTrustStoreFiletype>jks</dsmlTrustStoreFiletype>
</DsmlSSLConfig>
```

> **Note:** Most of the configuration of the clients is performed via command line at runtime. The only option configurable before runtime is the classpath, the requirements for which are listed above in the INSTALLATION section.

Execution

In this section we will discover how to execute IBM DSML service.

Server

Execution of the server is broken up into the SOAP binding component and the file binding component. The DMSL request is transformed into a JNDI call and it does all of the request for you to the LDAP server via JNDI.

SOAP binding

After DSML server has been deployed within SOAP, requests are made against it by submitting them to the SOAP webapp's messagerouter servlet. For Tomcat users, this URL should be similar to
`http://host:port/soap/servlet/messagerouter`. Simply submit a SOAP request to this URL, and you should receive a corresponding SOAP response. For the DSML SOAP component to function, the message files must be present in SOAPHomeDir\msg.

File binding

Execution of the server via file binding is achieved by invoking the DsmlFileClient with an XML input file containing a batchRequest element. To execute the DsmlFileClient, the message files must be present in a directory listed in the CLASSPATH.

> **Note:** The DSMLSSLConfigFileSchema.xsd file must be in the %JAVA_HOME%/jre directory. (This is done by default during installation.)

Clients

There are two types of DSML Clients applications in IBM DSML Service: DsmlSoapClient and DsmlFileClient.

DsmlSoapClient

Description: Used to submit a DSML request via SOAP to a DSML server deployed within a SOAP webapp.

Syntax:

```
java com.ibm.ldap.dsmlClient.DsmlSoapClient LDAP-userid LDAP-password [OPTIONS]
```

Options:

- ► -d debug - specified level of tracing (logically OR'ed bit flags - add numbers of desired debug levels)
    - – ERROR = 1
    - – WARNING = 2
    - – DIAGNOSTIC = 4
    - – INFORMATION = 8
- ► -i inputFileName - File containing DSMLrequest
- ► -l logFileName - Log file name (full path)
- ► -o outputFileName - File to contain DSML response
- ► -S DSMLSOAPserver - URL to SOAP messagerouter
- ► -s LDAP Server URL - LDAP server to which the DSML request applies

Defaults:

- ► debug = 3 (WARNING + ERROR)
- ► LDAP Server URL = DSML server default
- ► logFileName = DsmlLog.txt in the current directory
- ► inputFileName = DsmlRequest.xml in the current directory
- ► outputFileName = DsmlResponse.xml in the current directory

LDAP user ID and password, as well as SOAP user ID and password must be supplied.

DsmlFileClient

Description: Used to submit a file bound DSML request via a DSML server residing on the same computer as DsmlFileClient.

Syntax:

```
java com.ibm.ldap.dsmlClient.DsmlFileClient LDAP-userid LDAP-password [OPTIONS]
```

Options:

- ► -d debug - specified level of tracing (logically OR'ed bit flags - add numbers of desired debug levels)
    - – ERROR = 1
    - – WARNING = 2
    - – DIAGNOSTIC = 4
    - – INFORMATION = 8
- ► -i inputFileName - File containing DSML request
- ► -l logFileName - Log file name (full path)
- ► -o outputFileName - File to contain DSML response

- ► -s LDAP Server URL - LDAP server to which the DSML request applies

Defaults:

- ► debug = WARNING + ERROR (3)
- ► inputFileName = DsmlRequest.xml in the current directory
- ► logFileName = DsmlLog.txt in the current directory
- ► outputFileName = DsmlResponse.xml in the current directory
- ► ldapServer = ldap://localhost:389/

LDAP user ID and password must be supplied.

See Example A-46 for an example of using DsmlFileClient to submit a DSML search request.

Example: A-46   Using DsmlFileClient to submit a DSML search request

```
java com.ibm.ldap.dsmlClient.DsmlFileClient LDAP-userid LDAP-password
```

The DSML request file can be AddRequest, DelRequest, ModifyRequest, SearchRequest.

Note that if you want the DsmlFileClient to connect to the LDAP server using SSL, the DSMLSSLConfig.xml file must be in a directory that is in your CLASSPATH. See the SSL section of this document for information about the format of the DSMLSSLConfig.xml file.

Note that you need to copy DSMLSSLConfigFileSchema.xsd from the dsml unzipped folder to C:\Program Files\Java\j2re1.4.1_05.

There are extra classpath settings needed to run the DsmlFileClient. The classpath needs the following dsml jar files: dsml.jar, etc. as the following C:\DSML\jars\auibase.jar;C:\DSML\jars\dsml.jar;C:\DSML\jars\regex4j.jar;C:\DS ML\jars\IBMLDAPJavaBer.jar;C:\DSML; and be sure to set the servlet.jar classpath: C:\DSML\servlet.jar.

See Example A-47 on how to set a classpath.

Example: A-47   Setting the CLASSPATH

```
Set classpath=.;C:\DSML\jars\auibase.jar;C:\DSML\jars\dsml.jar;
C:\DSML\jars\regex4j.jar;
C:\DSML\jars\IBMLDAPJavaBer.jar;
C:\DSML;
C:\Progra~1\IBM\LDAP\appsrv\lib\j2ee.jar;
C:\PROGRA~1\IBM\LDAP\appsrv\lib\jaf-1.0.2\activation.jar;
C:\PROGRA~1\IBM\LDAP\appsrv\lib\javamail-1.3.1\mail.jar;
C:\PROGRA~1\IBM\LDAP\appsrv\lib\xml4j-4_2_2\XercesImpl.jar;
```

```
C:\PROGRA~1\IBM\LDAP\appsrv\lib\xml4j-4_2_2\XMLParserAPIs.jar;
C:\Progra~1\IBM\LDAP\appsrv\lib\soap.jar;
C:\PROGRA~1\IBM\SQLLIB\java\db2java.zip;
C:\PROGRA~1\IBM\SQLLIB\java\db2jcc.jar;
C:\PROGRA~1\IBM\SQLLIB\java\db2jcc_license_cu.jar;
C:\PROGRA~1\IBM\SQLLIB\bin;
C:\PROGRA~1\IBM\SQLLIB\java\common.jar;
```

You should be able to see the result of running the **rundsmlfileclient** command as being similar to Example A-48.

Example: A-48   Output from the rundsmlfileclient command

```
C:\Program Files\IBM\LDAP\idstools\DSML>rundsmlfileclient

C:\Program Files\IBM\LDAP\idstools\DSML>java
com.ibm.ldap.dsmlClient.DsmlFileClient cn=root manager
[Info] CSA Toolkit Version 1 Release 3
(c) Copyright IBM Corporation 2001, 2003. All rights reserved.

Using:
        Input file name: DsmlRequest.xml
        Output file name: DsmlResponse.xml
        LDAP server: ldap://localhost:389/


Finished processing DSML BatchRequests.
Results are in file: DsmlResponse.xml
```

DSMLRequest.xml

See Example A-49 for an example of a DSMLRequest.xml that can be used for DSML client to issue the request.

Example: A-49   DSMLRequest.xml for DSML client to issue a request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core">
   <dsml:batchRequest>
      <dsml:addRequest dn="cn=chunhui Yang, o=ibm.c=us">
         <dsml:attr name="objectclass">
            <dsml:value>person</dsml:value>
         </dsml:attr>
         <dsml:attr name="objectclass">
            <dsml:value>top</dsml:value>
         </dsml:attr>
         <dsml:attr name="cn">
            <dsml:value>chunhui yang</dsml:value>
```

```
            </dsml:attr>
            <dsml:attr name="sn">
                <dsml:value>yang</dsml:value>
            </dsml:attr>
        </dsml:addRequest>
    </dsml:batchRequest>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Troubleshooting

This section covers some common errors and causes that you may encounter during IBM DSML v2 implementation.

► Error: Exception in thread "main" java.lang.NoClassDefFoundError: org/w3c/dom/Node

  Cause: XMLParserAPIs.jar missing from CLASSPATH or JAVA_HOME/jre/lib/ext

► Error: Exception in thread "main" java.lang.NoClassDefFoundError: com/ibm/jsse/IBMJSSEProvider

  Cause: ibmjsse.jar missing from CLASSPATH or JAVA_HOME/jre/lib/ext

► Error: Exception in thread "main" java.lang.NoClassDefFoundError: org/apache/soap/util/xml/XMLParserUtils

  Cause: soap.jar missing from CLASSPATH or JAVA_HOME/jre/lib/ext

► Error: Exception in thread "main" javax.xml.transform.TransformerFactoryConfigurationError: Provider org.apache.xalan.processor.TransformerFactoryImpl not found

  Cause: xalan.jar missing from CLASSPATH or JAVA_HOME/jre/lib/ext

► Error of DSMLv2.xsd:

  The Error message will be like the one shown in Example A-50.

Example: A-50   DSMLv2.xsd error message

```
[Error] Fri Dec 12 21:23:30 EST 2003 Error at File:
file:///C:/Program%20Files/IBM/LDAP/idstools/DSML/DsmlRequest.xml
[Error] Fri Dec 12 21:23:30 EST 2003    Line: 1
[Error] Fri Dec 12 21:23:30 EST 2003    Column: 79
[Error] Fri Dec 12 21:23:30 EST 2003 cvc-elt.1: Cannot find the declaration of
element 'SOAP-ENV:Envelope'.
[Error] Fri Dec 12 21:23:30 EST 2003 org.xml.sax.SAXParseException: cvc-elt.1:
Cannot find the declaration of element 'SOAP-ENV:Envelope'.
```

```
        at
org.apache.xerces.util.ErrorHandlerWrapper.createSAXParseException(Unknown
Source)
    at org.apache.xerces.util.ErrorHandlerWrapper.error(Unknown Source)
    at org.apache.xerces.impl.XMLErrorReporter.reportError(Unknown Source)
    at org.apache.xerces.impl.XMLErrorReporter.reportError(Unknown Source)
    at org.apache.xerces.impl.xs.XMLSchemaValidator.handleStartElement(Unknown
Source)
    at org.apache.xerces.impl.xs.XMLSchemaValidator.startElement(Unknown
Source)
    at org.apache.xerces.impl.XMLNSDocumentScannerImpl.scanStartElement(Unknown
Source)
    at
org.apache.xerces.impl.XMLNSDocumentScannerImpl$NSContentDispatcher.scanRootEle
mentHook(Unknown Source)
    at
org.apache.xerces.impl.XMLDocumentFragmentScannerImpl$FragmentContentDispatcher
.dispatch(Unknown Source)
    at
org.apache.xerces.impl.XMLDocumentFragmentScannerImpl.scanDocument(Unknown
Source)
    at org.apache.xerces.parsers.XML11Configuration.parse(Unknown Source)
    at org.apache.xerces.parsers.DTDConfiguration.parse(Unknown Source)
    at org.apache.xerces.parsers.XMLParser.parse(Unknown Source)
    at org.apache.xerces.parsers.DOMParser.parse(Unknown Source)
    at org.apache.xerces.jaxp.DocumentBuilderImpl.parse(Unknown Source)
    at javax.xml.parsers.DocumentBuilder.parse(DocumentBuilder.java:124)
    at com.ibm.ldap.dsmlClient.DsmlSoapClient.main(DsmlSoapClient.java:205)
```

Causes: This is because in the xsd document, it should use single quote (')
instead of double quote (") as shown in Example A-51.

Example: A-51   xsd document quote problem

```
……
      <xsd:pattern value='[0-2]\.[0-9]+(\.[0-9]+)*'/>
   </xsd:restriction>
……
```

► Known issues:

The DSML file client throws a java exception similar to Example A-52 on
page 673 when it is set up to communicate using SSL and the user tries to
connect to an LDAP server that does not use SSL.

Example: A-52   SSL Java exception

```
SSL IS ON javax.naming.CommunicationException: 9.182.21.228:389. Root exception
is javax. net.ssl.SSLProtocolException: end of file
    at com.ibm.jsse.bd.a(Unknown Source)
```

```
    at com.ibm.jsse.b.a(Unknown Source)
    at com.ibm.jsse.b.write(Unknown Source)
    at com.sun.jndi.ldap.Connection.<init>(Connection.java:226)
    at com.sun.jndi.ldap.LdapClient.<init>(LdapClient.java:127)
    at com.sun.jndi.ldap.LdapCtx.connect(LdapCtx.java:2398)
    at com.sun.jndi.ldap.LdapCtx.<init>(LdapCtx.java:258)
    at
com.sun.jndi.ldap.LdapCtxFactory.getInitialContext(LdapCtxFactory.java:91)
    at javax.naming.spi.NamingManager.getInitialContext(NamingManager.java:674)
at javax.naming.InitialContext.getDefaultInitCtx(InitialContext.java:255)
    at javax.naming.InitialContext.init(InitialContext.java:231)
    at javax.naming.InitialContext.<init>(InitialContext.java:207)
    at
javax.naming.directory.InitialDirContext.<init>(InitialDirContext.java:92)
    at com.ibm.ldap.dsml.DsmlRequest.processRequests(DsmlRequest.java:767)
    at com.ibm.ldap.dsml.DsmlServer.processDsmlRequest(DsmlServer.java:253)
    at com.ibm.ldap.dsml.DsmlServer.processDsmlRequest(DsmlServer.java:402)
    at com.ibm.ldap.dsml.DsmlServer.processDsmlRequest(DsmlServer.java:373)
    at com.ibm.ldap.dsml.DsmlServer.processDsmlRequest(DsmlServer.java:296)
    at com.ibm.ldap.dsmlClient.DsmlFileClient.main(DsmlFileClient.java:203)
```

The exception is not fatal and the output XML file is generated.

Java programming examples on DSML

DSML v2 integrates XML and directories by providing the ability to perform
nearly all LDAP operations in XML. DSML v2 operations are purely based on
LDAP counterparts, so LDAP information and operations can be directly mapped
to DSML v2 operations.

As we explained earlier, on the server side, each DMSL request is transformed
into a JNDI call and it does all of the request for you to the LDAP server via JNDI;
on the client side, an xml application submits DSML requests cross HTTP
network. In this section, we provide some sample programming codes to show
you how to manipulate DSML request in Java. 4.2.1–4.2.3 are some sample java
programming codes for DSML client, 4.2.4–4.2.6 are some sample java
programming codes to parse dsml document, execute dsml query both in LDAP
and SOAP, and retrieve result back on the server.

JNDI introduction

JNDI provides a standard interface to various directory and name services
through the extensible provider architecture, it offers powerful ability to perform
all the important LDAP operations from Java. The benefit of using DSML and
JNDI to access a server that is providing DSML information via HTTP is that you

don't need to expose information natively with LDAP in a server, especially in an enterprise environment, it is almost impossible to expose its directory information.

JNDI functionality is under javax.naming hierarchy, javax.naming classes are used to handle simple name services, and javax.naming.directory classes are extended and used to handle complex directory service.

The common two JNDI drivers for LDAP are:

► Sun LDAP provider for JNDI: com.sun.jndi.ldap.LdapCtx.Factory, it is part of main J2SE distribution.

► IBM LDAP provider for JNDI: com.sun.jndi.ldap.LdapCtx.Factory, it is part of IBM directory server client development kit.

All JNDI operations are performed by DirContext object which connects to LDAP server and then proceeds the JNDI operations. The connection has to be closed, and a NamingException will be thrown for you to handle.

Program examples

This section covers some of the basic Java programming on DSML manipulation with the latest DSML v2 operation capabilities. It covers from creation of a connection, creation of DSML document on the fly, sending DSML request and retrieving DSML response on the client side, to parsing of DSML document, converting DSML request to JNDI LDAP request, and retrieving JNDI LDAP operation result back on the server side.

DSML Client - Create the connection

Example A-53 shows you how to create the URL connection of the DSML server.

Example: A-53   DSML Client - Create the connection

```
……
   url = new URL(DsmlURL);
   connection = url.openConnection();
   connection.setRequestProperty ("Authorization", "Basic " + encoding);
   httpConn = (HttpURLConnection) connection;
……
```

DSML Client - Set the HTTP parameters

Example A-54 shows you how to set the HTTP parameters for the dsml request.

Example: A-54   DSML Client - Set the HTTP parameters

```
……
```

```
httpConn.setRequestProperty( "Content-Length",
    String.valueOf( b.length ) );
httpConn.setRequestProperty("Content-Type","text/xml; charset=utf-8");
httpConn.setRequestProperty("SOAPAction",SOAPAction);
httpConn.setRequestMethod( "POST" );
httpConn.setDoOutput(true);
httpConn.setDoInput(true);
```
……..

DSML Client - Generate DSML document

If you need to generate DSML request document on the fly, you can use either
Document Object Model (DOM) or Simple APIs for XML (SAX) which are
interfaces to dynamically access and update document content.

See Example A-55 for an example of generating DSML document.

Example: A-55   DSML Client - Generate DSML document

……
```
//build new DOM document
DoucmentBuilder builder=factory.newDocumentBuilder();
dsmlDoc=builder.newDocument();
```
……
```
//create dsml batchRequest element
Element br=(Element) dsmlDoc.createElement("dsml:batchRequest");
br.setAttribute("xml:dsml", "urn:oasis:names:tc:DSML:2:0:core");
//create add request which is the child element of batchRequest element
Element ar=(Element) dsmlDoc.createElement("dsml:addRequest");
ar.setAttribute("dn", dn);
ar.appendChild(dsmlDoc.createTextNode("\n"));
br.appendChild(ar);
```
……
```
//create other dsml elements
```
……

ITDS DSML service does not require you to code anything to generate DSML
document, all you need to do is to create a dsml document in a editor, and
specify the dsml file name in the DSML client application commands.

DSML Client - Get DSML servlet response

Example A-56 shows you how to read the response returned from the DSML
servlet.

Example: A-56   DSML Client - Get DSML servlet response

……
```
InputStreamReader ireader =
```

```
     new InputStreamReader(httpConn.getInputStream());
…….
```

DSML Servlet - Parse DSML document

The steps to Parse a DSML document is shown in Example A-57 (using SAX).

Example: A-57   DSML Servlet - Parse DSML document

```
   //create a new instance
   SAXParserFactory myIns=SAXParserFactory.newInstance();
   myIns.setNameSpaceAware(true);
// This is because SAX complains an expression error about the
// http://www.w3.org/2001/XMLSchema.xsd.
   myIns.setValidating(false);

   //create new SAX Parser
   myParser=myIns.newSAXParser();

   //Parse dsml file
   SchemaXMLHandler myHandler = new SchemaXMLHandler();
   MyParser.parse(dsmlfile, myHandler);
```

For detail method specification, please refer to:

```
http://java.sun.com/j2se/1.4.2/docs/api/javax/xml/parsers/SAXParser.html#pa
rse(java.io.File,%20org.xml.sax.helpers.DefaultHandler)
```

DSML Servlet - JNDI DSML search

In order to perform DSML Search operation in JNDI, you have to set the DSML provider context factory, LDAP URL information that DSML Search operation is querying, LDAP credentials (userid and password), and then retrieve search return entries in DSML format. Example A-58 shows the java programming code to do this.

Example: A-58   DSML Servlet - JNDI DSML search

```
……
   // set DSML provider context factory
   Hashtable DsmlEnv=new Hashtable();
   DsmlEnv.put(Context.INITIAL_CONTEXT_FACTORY,
       "com.sun.jndi.dsml.DsmlCtxFactory");

   // set LDAP URL and query base
   DsmlEnv.put(Context.PROVIDER.URL, "ldap://localhost" + base + "?"
                 + attrs + "?" + scope +"?" +filter);

   // set LDAP credentials
```

```
DsmlEnv.put(Context.SECURITY_PRINCIPAL, "cn=root");
DsmlEnv.put(Context.SECURITY_CREDENTIAL, "password");

//create DirContext object to retrieve query results
DirContext DsmlCtx=new InitialDirContext(DsmlEnv)
String DsmlResult=DsmlCtx.lookup("").toString();
DsmlCtx.close();
......
```

The DSML search result will be in dsml:directory-entries element as shown in Example A-59, which contains multiple dsml:entry elements if there are multiple entries returned in the query result.

Example: A-59   DSML search result in dsml:directory-entries

```
<dsml:directory-entries>
    <dsml:entry dn="entry1">
        <dsml:objectclass>
            <dsml:oc-value>person</dsml:oc-value>
            <dsml:oc-value>top</dsml:oc-value>
        </dsml:objectclass>
        <dsml:attr name="cn">
            <dsml:value>entry1</dsml:value>
        </dsml:attr>
        ......
    </dsml:entry>
    <dsml:entry dn="entry2">
        <dsml:objectclass>
            <dsml:oc-value>person</dsml:oc-value>
            <dsml:oc-value>top</dsml:oc-value>
        </dsml:objectclass>
        <dsml:attr name="cn">
            <dsml:value>entry2</dsml:value>
        </dsml:attr>
        ......
    </dsml:entry>
    ......
</dsml:directory-entries>
```

DAML Servlet - JNDI create DSML SOAP request

To use JNDI to create a DSML SOAP request is pretty much the same with the previous JNDI DSML Search operation, the only difference is the context factory. See Example A-60 for an example of using JNDI to create a DSML SOAP Request. By defining the JNDI DSML SOAP context factory, any further JNDI requests on the dirContext object would be performed in DSML instead of LDAP.

Example: A-60   DAML servlet - JNDI create DSML SOAP request

```
……
// set DSML provider context factory
   Hashtable DsmlEnv=new Hashtable();
   DsmlEnv.put(Context.INITIAL_CONTEXT_FACTORY,
"com.sun.jndi.dsmlv2.soap.DsmlSoapCtxFactory");

   // set LDAP URL and query base
   DsmlEnv.put(Context.PROVIDER.URL, "ldap://localhost:8080");

   //retrieve query results
   DirContext dirContext=new InitialDirContext(DsmlEnv);
   String DsmlResult=DsmlCtx.lookup("").toString();
   DsmlCtx.close();
```

DSML Servlet - JNDI operations

Like the JNDI Search operation we explained above, other JNDI operations such as modify, add can be performed in the similar way using corresponding methods of DirContext object. For more information on DirContext object methods and Context object methods, please refer to Sun's Web site located at:

http://java.sun.com/products/jndi/1.2/javadoc/javax/naming/directory/DirContext
.html

http://java.sun.com/products/jndi/1.2/javadoc/javax/naming/Context.html

References to the DSML official specifications

► ITDS 5.2 documentation

  http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html

► *Using LDAP for Directory Integration*, SG-6163

  http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246163.h
  tml?Open

► *Implementation and Practical Use of LDAP on the IBM eServer iSeries Server*, SG24-6193

  http://publib-b.boulder.ibm.com/Redbooks.nsf/9445fa5b416f6e32852569ae006
  bb65f/219b250894a046e285256b11006da9d9?OpenDocument&Highlight=0,ldap

► DOM documentation

  http://www.w3.org/DOM/

► JNDI documentation

  http://java.sun.com/products/jndi/1.2/javadoc/

- ► SAX

  `http://www.saxproject.org/`

Directory Integration - IBM Tivoli Directory Integrator

In this appendix we discuss the topic of directory integration and the IBM Tivoli Directory Integrator (ITDI). ITDI is a toolkit that simplifies integration of directories with a variety of other data stores. It includes "snap-in" components that support connectivity to many types of data stores, protocols for the reliable and secure transportation of data between them, and help automate the mapping of data between them. ITDI includes "event handlers" to help systems integrators automate data flows between data stores in near real time as data is changed or based on defined schedules. The ITDI architecture includes a framework for scripting business rules as data is exchanged between repositories. It even has special "plug-ins" for a variety of directories that enable it to securely capture and propagate passwords when they are changed. These include Active Directory, the iPlanet directory, the IBM Directory Server, and the HTTP password on Domino Servers.

Before we dive into how to use IBM Tivoli Directory Integrator for directory integration, we need to clarify why the topic is a critical success factor in enterprise directory projects so often that it deserves a chapter in this book. After all, directories based on the LDAP standards are now available from many vendors. They are embedded with modern operating systems, Internet security systems and application servers. Why is a directory solution not complete after a highly available LDAP directory service is in place with a well-designed directory tree and a replication strategy that delivers high availability and performance?

Would the final step not be to enforce a corporate edict that all infrastructure services and applications use that directory service to maintain and access the identity and other data that it contains?

Why Directory Integration is important

Today nearly all modern network operating systems[1], I/T infrastructure components, and applications that require user identity data are directory enabled "off the shelf." They can exploit an LDAP directory for user authentication and access control as soon as they are installed and properly configured. But the reality is that this technology has emerged only recently and was standardized over just the last few years. In the not too distant past, enterprise applications were often installed with their own application specific repositories to maintain the identities of users, their preferences and permissions to access application resources.

As the Internet was embraced to extend access to corporate IT resources and applications to customers and business partners, security software and Web application servers also needed a trusted repository for user identity data. As a result, most enterprises have in place many legacy repositories of identity data. Some of them may be based on modern standards and access protocols and others may not. Some may contain only internal users while other databases contain the data used to authenticate business partners and customers who access corporate Web sites. But there is a considerable investment in business processes and applications that keep these repositories populated with data and their contents up to date as users are added and removed, or their business roles change and affect access permissions to application functions and data.

The problem is not that most enterprises want another application specific repository but that they have too many already with maintenance responsibilities for them split between several departments with different, and sometimes conflicting, business objectives. The investments enterprises have in Human Resource (HR) systems, application databases, legacy application and LAN user directories means that those systems are not easily replaced and will remain the "authoritative sources" where the identity data for users of corporate applications is maintained for some time. This is also the case for the basic identity data for business partners and customers who use new Web-based applications to access corporate resources. The result is that in most directory deployments, corporate or enterprise LDAP directories do not immediately become the "authoritative source" for most of the data they contain. They need to become instead "trusted sources" that provide infrastructure components and applications a standard mechanism to access identity and other shared data for user authentication and access control. The data the corporate LDAP directory contains will mostly originate and be maintained elsewhere and must be kept

---

[1] While modern operating systems are directory enabled, they are not always installed and configured to exploit an LDAP directory for user accounts. For example, Unix and Linux servers are still configured to use flat files to maintain their user account and group information more often then they are set up to store user account information in a directory.

up-to-date with several authoritative sources. These problems are shown in
Figure B-1.
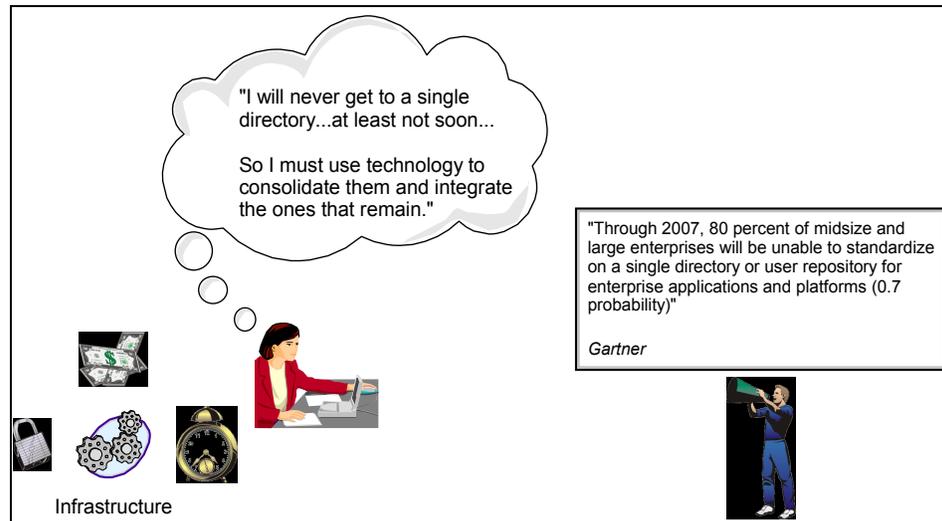
*Figure B-1   Cost, security policy, complexity, and schedules drive requirements for
directory integration*

Directory Integration Services

So, by itself, the LDAP directory service installed with modern network operating
systems, security and reduced sign-on services, or Web services applications
will have limited business value to owners of legacy applications. Until it is
connected to existing repositories, the information the LDAP directory contains
about users and other IT resources will be stale, inaccurate, or inconsistent with
user accounts in operating systems, mail systems, and databases used by line of
business applications. The costs of revising legacy applications to exploit a new
directory service and changing the current administration applications and
procedures in several departments to maintain it is usually too high to offer a
satisfactory return on investment. The lengthy project schedules can mean the
directory services would be delivered too late to be of real value.

An alternative approach that extends the benefit of an enterprise directory
immediately is to use directory integration services[2] to reduce the number of
application specific directories and the maintenance of redundant data stores. It
may not be possible to completely eliminate application specific repositories or

---

[2] The term "directory integration services" is used to label various types of services that link
heterogeneous repositories and provide a consistent view of the information they contain. Another
popular label for these services is "metadirectory", which is a particular technology to accomplish this.

directories, but if they can be integrated and synchronized with each other, most of the legacy investment in maintenance of user identity data can be preserved and leveraged to populate and maintain an enterprise directory. The reduced maintenance costs and elimination of security exposures that are obtained by consolidating legacy directories[3] and eliminating redundant data also offers a compelling return on investment for directory integration projects.

User provisioning applications

User provisioning applications automate the management of user accounts for employees, business partners, and customers. These have traditionally been paper-based and process-intensive tasks that are often slow and error prone. Provisioning applications provide administrative interfaces and support workflows within the enterprise associated with provisioning and de-provisioning user accounts on operating systems, email and other application directories. They also manage the permissions individual users have to access corporate resources and IT transactions. They streamline and automate these procedures, ensuring that business policies are enforced and an audit trail is maintained.

Automated provisioning is often an objective of many corporate directory and security projects for the reasons illustrated in Figure B-1 on page 684. Information Week did a study where they interviewed 4500 security professionals and asked them whom they suspected as the source of a security breach or espionage. Six of the top eight suspects in this survey were people who were actually granted accounts with the permissions needed to perform the activity by the enterprise itself. The fact that 30–60 percent of the active accounts in an enterprise may be orphans[4] is obviously part of the problem. Perhaps the account should never have been created; maybe the user left the company or was promoted or transferred to a job with responsibilities that no longer require the account. The user may even have been a contractor whose contract has expired but who was never taken out of operating system or application directories.

Enterprises gain control of identity data by deploying provisioning systems. The security risks of orphan accounts are eliminated, and the business policies that govern the access permissions granted to the accounts are enforced. Accounts are turned on only for business purposes and are immediately turned off when the business reason for them is no longer valid.

---

[3] An internal study found IBM spends approximately $50 per user entry for each password directory it maintains. In a company with 200,000 employees, that offers significant savings in ongoing costs for each directory that can be consolidated or eliminated.

[4] An orphan account is one that exists and is associated with access permissions but the individual who owns the account is not known. Orphan accounts are obviously fertile ground for security problems.

However, a provisioning system is dependent on an underlying directory service or other database that is a trusted source of identity information about users and their enterprise roles. This identity data governs account management and the permissions that individual accounts are granted. The accuracy and currency of this identity data is crucial to the effectiveness of automating and enforcing enterprise account provisioning policies. You have to have a trusted source of information about who the valid users are and the enterprise roles and responsibilities they have before you can safely automate account provisioning and the assignment of permissions to accounts to execute transactions and access data[5]. A solid and well-integrated directory infrastructure is prerequisite to deploying a provisioning system.

Directory Integration technologies

With this background in why the directory integration is a crucial aspect of nearly every enterprise directory strategy, we should look at some of the different methods and technologies that have emerged to accomplish it. One solution that was used prior to the availability of tools better fitted to the purpose is to write scripts or small applications that synchronize information in one repository with another. This seems reasonable if all that is required is to synchronize a few data elements, or attributes, in a new enterprise directory with a single legacy system. No new infrastructure software need be purchased. If the synchronization does not need to be done immediately, it is usually possible to extract a few attributes from the source and periodically schedule a utility or application to update the directory.

But requirements are rarely that simple, and this approach can result in a solution that is error prone and difficult or impossible to maintain or extend over time. If we look at the tasks that must be accomplished to integrate heterogeneous repositories of data, it is easy to see why.

A robust directory synchronization strategy will usually require managing data flows between several systems. The solution design must incorporate a large amount of detail about bidirectional exchanges of information. The direction of updates must be consistent with the authoritative sources, so that data flows only from authoritative to non-authoritative sources. There are often requirements to detect changes that are made by a non-authoritative source, log the event, and possibly even rolling them back. Figure B-2 on page 687 illustrates many of the details that need to be understood and handled.

---

[5] A provisioning system that automates account creation and access permissions based on erroneous user identity data may be worse and actually less secure than an inefficient manual one.
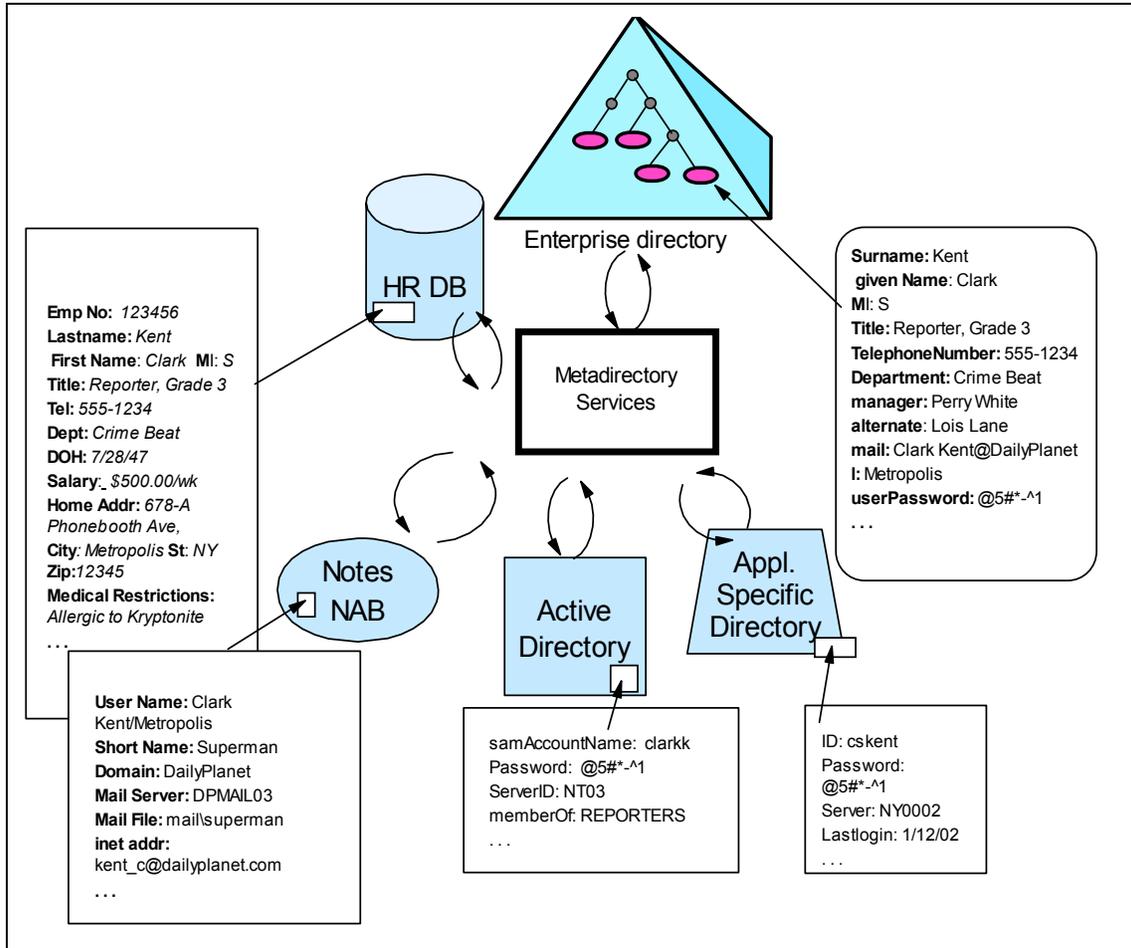
Figure B-2   Directory Integration requires handling many details

Data sources

Data sources are the systems and devices that are to communicate with each other.   Since a repository may be authoritative for only some of the data it contains, and receive updates from other systems, the data flows are often bidirectional.   For example, an HR system is usually authoritative for basic identity information of users such as name, address, the department to which an employee is assigned and their manager. An email or network operating system may receive information needed to create or delete accounts from the HR system, but be authoritative for other data such as login IDs and the groups to which users are assigned to provide access to network and other resources.

Matching rules

The chief goal of directory integration is to associate and synchronize data across various applications and data stores that use it. In doing so, the enterprise separates the management of shared (or enterprise) data from applications that work with it and maintain it. This is the essential first step in controlling identity data as a valuable corporate asset. Applications and infrastructure components such as security systems become consumers of corporate identity data, though legacy applications may remain the authoritative sources of the enterprise data.

However, it is not always easy to match up and associate the entries in one directory with those in another. For example, there may be no unique identifier, such as an *employee* or *customer number* for each user within an enterprise. The enterprise may have more than one HR system that assigns such numbers to employees in different divisions that have been acquired over time. And even if there is a unique identifier for each employee, it may not be currently stored as an attribute of the employee's entry in each of the directories and databases, and operating system platforms for which he or she has an account.

In Figure B-2 on page 687 for example, Clark Kent's key in the HR system is *employee Number* 123456. In Active Directory, Clark is assigned the *sAMAccountName* clarkk. He uses the *shortName* Superman in the corporate email system. The application specific directory may simply list two or three individuals with the name Clark Kent who have different user *IDs*. The only way to distinguish between them may be to look at the department they currently work in, or perhaps their office number or telephone numbers, and even this may not be enough to reliably associate them with our super hero. And, as we saw in the previous section if this is the organization's first directory project, there is a high probability that each of these systems may contain "orphan" accounts that cannot be matched with any individual or accounts in other repositories.

Entry and Attribute filtering and mapping

Each repository may contain information about many types of users or other resources. Some of this information is application specific or sensitive information that is stored in entries that business or privacy rules require should never be propagated to other systems. Within individual entries, there may be sensitive fields whose values should never be exposed while other attributes of the same entry must be frequently propagated to multiple destinations.   For example, salary and medical information that is contained in the HR database should never be propagated to other directories while first name, last name, and department are propagated to several other systems. The requirements for which fields are propagated change periodically as new applications are added and business rules change, so the solution must be easy to re-configure to adapt.

Attributes will often have different names and syntaxes in source and target systems. In the figure for example, *last name* in HR is mapped to *surname* (sn) in the enterprise directory.   Some attribute values for a target system don't have a direct mapping and may have to be computed from values in one or more source systems, as when *first name*, *middle initial*, and *last name* in an HR system are combined to create a *cn* (common name) attribute in an LDAP directory. This is a very simple example, but attribute mapping rules can be much more complex. For example, the users in LDAP directories are organized into a hierarchical directory tree with a distinguished name (DN) that specifies the precise location of their entry in the tree. When groups are synchronized between directories with different tree structures the groups contain the DNs of users in a *member* attribute. The DNs have to be mapped between the tree structures as the group entries are copied or synchronized between the directories.

Mechanisms to transport data between systems

Various standard protocols are used to transport data from source systems to a target systems. For some legacy systems, there is no standard protocol available to access or update the data they contain, and the systems integrator must use application specific programming interfaces or utilities provided by a vendor to import and export data. The data flows must be reliable, so that information is not lost due to system or network failures. Message queueing systems that guarantee delivery may have to be exploited.

When source and target systems communicate

Sometimes the systems need only be synchronized periodically, such as once a day or once an hour. This is often satisfactory when adding new hires into an employee database that will not actually become active or need access to systems right away. Other times, for example, when a user account must be revoked because of a possible security breach, or a user changes their password in the network operating system and it has to be propagated securely to other directories or applications, the communication must occur in near real time.

Change detection

Many systems have built-in mechanisms an integration system can exploit to detect changes. Directory servers have time stamps on the entries they contain. They can be configured with change logs and most support event notification or persistent search operations that enable a client application to register and receive modified entries when they are changed. Databases support triggers and stored procedures that can be exploited to detect and propagate changes. This is not always the case with legacy systems. For example, if the Application Specific Directory illustrated in the figure below is a spreadsheet or flat file, it is not so straightforward to look at a new version of the file and determine what entries

have been added, deleted, or modified since the last version, and which have not changed.

Logging and error handling

There are usually requirements to log certain transactions and events in system logs and to interface with systems management tools when failures occur within the integration solution. External systems or network failures may prevent synchronization from occurring at the time changes are made in a source system or when they are scheduled to be propagated. The integration solution must be robust enough to find these changes and propagate them correctly when the error is corrected and connectivity is restored. Failures may occur during a batch process that synchronizes a large number of entries and checkpoint/restart capabilities may be required.

Metadirectories and virtual directories

Due to the complexity of these requirements, custom scripting or application development is not usually affordable or maintainable. It is viable only for solutions that involve only a few point-to-point data flows with minimal requirements for event handling, attribute mappings and minimal logging and error handling capabilities. Metadirectories are tools that have emerged to provide a complete set of services tailored to handling these issues. They enable integrators to quickly develop, deploy and maintain and extend a solution for integrating identity data for infrastructure components and applications.

A metadirectory is not another user directory. It is a toolkit that provides graphical tools systems integrators use to work with information about where data is located, how it can be accessed, how the entries in one store are linked with entries in another, and how the data should flow between different directories and databases. Metadirectory run-time services include connectors (agents) for collecting information from many operating system and application specific sources to integrate the data into a unified namespace.   Metadirectories also enforce business rules that specify the authoritative source for attribute values, handle naming and schema discrepancies, and provide data synchronization services between information sources. One of the benefits of a metadirectory is that it can create and maintain a central repository consisting of entries and attributes that are "joined" or aggregated from many other sources. However, a central store for data other than the "metadata" is not required for a metadirectory to provide synchronization services.[6]

---

[6] Some metadirectory vendors require that a proprietary product or database be used to store the metadata. This is not a characteristic of ITDI. ITDI stores its static metadata in XML configuration files and can store dynamic metadata in an embedded java database.

Virtual directories vs. metadirectory technology

Virtual directories implement relatively new, but closely related and complementary to metadirectory technology with similar services. They provide applications with "virtual views" of the data contained in a variety of data stores. These views can be tailored to the requirements of the application. An application that prefers to use LDAP protocol to access its data can do so, even though the data may be stored in a relational database. Virtual directories are essentially brokers that enable a single query to reference information in multiple data sources dynamically.

A virtual directory could assemble information from multiple sources, perhaps using attributes in a directory as pointers, and then present it to a client application in response to an LDAP query against a virtual directory tree that is defined in the virtual directories "metadata". In Figure 3, a virtual directory could receive a query for Clark Kent's data and assemble his phone number, e-mail address, Active Directory and Lotus Domino account IDs, and his physical address from the multiple directories to enable an application to present them on a single screen for centralized administration. A virtual directory provides a layer of abstraction between the applications accessing data and the various repositories where it is stored and managed.

A potential advantage of a virtual directory over a metadirectory, when data access is primarily read-only and there is no need to synchronize data at the various sources, is that data is not moved between sources in order to compose and permanently store an aggregate view. Instead, the data is aggregated as required by the applications that access it. Virtual directories could be appropriate when this is the fundamental requirement, rather than data synchronization, especially for large amounts of data that is mostly read and infrequently written.

In many situations the advantage of a virtual directory can be very difficult to achieve. Directories and databases achieve high performance for portals and security systems that must perform hundreds of authentications and other queries on directory data per second by caching data.   Since they control all access to the data, the directory server or database engine can manage a cache efficiently by discarding or replacing cached data when updates are made. Virtual directories can also cache data, of course, but a highly efficient caching strategy is more difficult for them because they do not see updates to the underlying data stores by applications that bypass them and write directly to the data store. When the virtual directory must store cached data persistently due to memory limitations on the server hardware or to provide quick restarts of the server, the distinction between virtual directories and metadirectories is blurred.

Since virtual directories synthesize views of information that can physically reside in several stores with different schemas, they will include most of the functionality of a metadirectory. For this reason, it is likely that metadirectories will evolve to provide some virtual directory services over time. The IBM Tivoli Directory Integrator may be an example of the very early stages of this trend. The latest release of ITDI includes an LDAP server "event handler" that enables a configuration to act as an LDAP server to a client application or infrastructure component. An ITDI configuration can therefore be built to receive LDAP requests and run assembly lines to perform operations such as looking up data in various types of data sources and assembling a resulting view of data in heterogeneous stores to a client using LDAP protocol.[7] It is likely that over the long term, both metadirectories and virtual directory approaches will have a role in directory integration.

Overview of IBM Tivoli Directory Integrator

IBM Tivoli Directory Integrator synchronizes identity data residing in directories, databases, and applications. By serving as a flexible, synchronization layer between a company's sources of identity data, Directory Integrator can eliminate the need for a centralized data store. For those enterprises that do need to deploy an enterprise directory, Directory Integrator can connect it to the identity data from various repositories throughout the organization to populate it and keep it up-to-date.

With many built-in connectors for managing the flow of data and events, an open-architecture Java development environment to extend or modify these connectors, and tools to apply logic to data as data is processed, Directory Integrator can be used for:

► Synchronizing and exchanging information between applications or directory sources.
► Managing data across a variety of repositories providing the consistent directory infrastructure needed for a wide variety of applications including security and provisioning.
► Creating the authoritative data bases needed to expose only trustworthy data to advanced software applications such as Web services.
► Building an Enterprise Directory that contains commonly used data about users. This data store can become the authoritative data store for these data elements for the enterprise.

The product architecture is illustrated in Figure B-3 on page 694. The key concept to understand about ITDI is that it implements an assembly line model in

---

[7] This is not yet a true virtual directory, of course. ITDI does not cache the virtual views created by the assembly lines and does not include tools specifically designed to create them.

which components are configured to support data flows between data sources. The product includes a GUI toolkit environment for creating and modifying assembly lines and a run-time server with a management GUI for starting and stopping assembly lines, examining log files, and other functions required to support an infrastructure service.

The basic components used to compose ITDI assembly lines are described in this section. Complete documentation, including a tutorial that guides the new user through the creation and operation of a simple assembly line is available in the on-line publications packaged with the product. The online documentation also includes a number of sample configurations that illustrate the use of the connectors and event handlers packaged with the product.

ITDI configurations are stored in XML files that contain all of the static metadata about one or more assembly lines. There is no architectural limit to the number of assembly lines that can be defined and managed by a single configuration file. An ITDI server is started with a particular configuration file, so an instance of the ITDI server can run several assembly lines concurrently. ITDI is a Java application and does not have dependencies on other infrastructure services.[8] It is possible to start multiple instances of the ITDI server on a single system, each with configuration files defining different sets of assembly lines.

---

[8] This is an important point. Some metadirectories have dependencies on other services such as a particular vendor's LDAP server or relational database being installed with it. With ITDI, the hardware and software requirements to install and use it are lightweight and depend mainly on having connectivity to the data sources it is used to integrate.
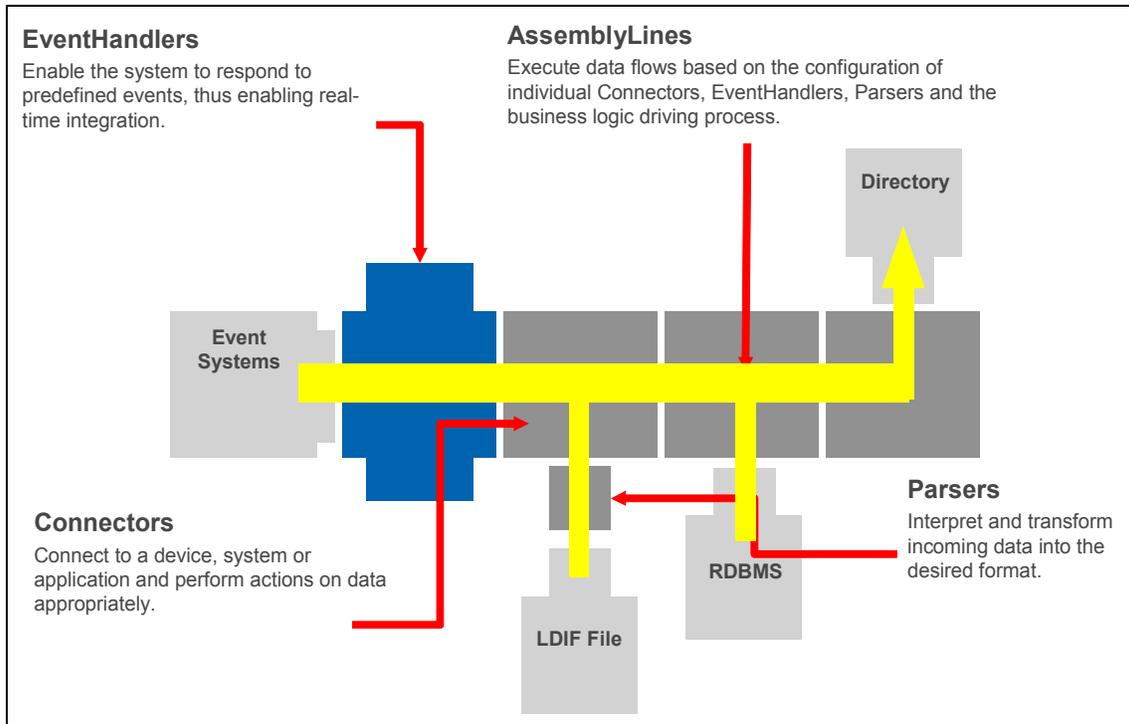
Figure B-3   IBM Tivoli Directory Integrator Product Architecture

Assembly lines

An *assembly line* collects information from connected information sources and performs operations on the data. An assembly line can create new entries altogether or update and delete entries in data sources. Assembly lines receive information from various data sources, perform operations on the input, and then map the collected information to other data sources. Assembly Lines work on one item at a time, for example, one data record, directory entry, registry key, and so forth. Multiple assembly lines may be running concurrently.

Connectors

Connectors support numerous protocols and access mechanisms. Many connectors are included with the product and ITDI architecture includes a framework that allows advanced users to create others. Connectors provide the input and output of an Assembly Line. Each Connector is linked to a data source, and is provides an environment specific to that type of source where data transformation and aggregation operations are easy to configure.

ITID connectors can be configured in several modes that determine how they operate. In *iterator* mode a connector is configured to read through the entries in a data source[9] and pass each entry down the assembly line. In *lookup* mode, a connector is configured to find an entry in a data source based on search criteria and retrieve data from it. In *update* mode, a connector is configured to lookup an entry and may also update the entry after it has been located. In *delete* mode, a connector is configured to find and remove an entry from a data source. In *add* mode, a connector is configured to add entries to a data source.

In the current release the built-in connectors include:

- ► Btree Object DB Connector
- ► Command Line Connector
- ► Domino Users Connector
- ► File System
- ► FTP Client Connector
- ► Old HTTP Client Connector
- ► HTTP Client Connector
- ► Old HTTP Server Connector
- ► HTTP Server Connector
- ► IBM MQ Series (JMS)
- ► IBM Directory Changelog Connector
- ► JMS Connector
- ► JNDI
- ► LDAP
- ► Lotus Notes
- ► MailboxConnector Connector
- ► Memory Stream Connector
- ► Netscape/iPlanet Changelog Connector
- ► NT4
- ► Script Connector
- ► SNMP Connector
- ► TCP Connector (Generic)
- ► URL Connector (Generic)
- ► (runtime provided) Connector
- ► Web Service Connector

Event Handlers

ITDI includes an Event Handler framework that provides the integrator the ability to wait for, recognize and react to specific events that take place in the infrastructure. For example, changes in an LDAP or NOS directory, arriving e-mails, records updated in certain databases, incoming HTML requests for pages from a Web browser or Web services-based Simple Object Access

---

[9] The iteration can be filtered. For example an LDAP or database connector in iterator mode can be configured to read through all the entries returned by specific search criteria.

Protocol (SOAP) messages can trigger assembly lines to run or other operations. The advanced user may also define new event handlers for site-specific events[10].

ITDI Event Handlers are configured through the configuration GUI to examine the event[11] and perform one or more actions based on the information received about the event. An action can be simple such as running a particular assembly line. More complex configurations for event handlers are also possible, such as selecting an assembly line to run based on data associated with the event, running a sequence of assembly lines, and executing scripts.

The built-in event handlers include:

► Active Directory Changelog.
► DSMLv2.
► Microsoft Exchange Changelog.
► HTTP.
► IBM Directory Server Changelog. This event handler is specifically designed to work efficiently with the Tivoli IBM Directory Server's changelog and event notification functionality.
► JMX.
► LDAP event handler for use with directories that support persistent search.
► LDAP server handler that enables ITDI to operate as an LDAP server to a client.
► SNMP.
► TCP Port.
► Timer.
► Web Service.
► zOS LDAP Changelog.

Parsers

Parsers interpret and translate information from a byte stream into a structured information object, where each piece of information is accessible by name. You can also translate a structured information object into a byte stream. You can select from the wide range of extensible parsers such as "comma separated values", "fixed column", LDAP Data Interchange Format (LDIF), Extensible Markup Language (XML), SOAP, and Directory Services Markup Language (DSML), or even create a new parser from scratch.

---

[10] For example, ITDI includes a script event handler that enables the integrator to write a script to define a custom event.
[11] As an example, ITDI includes LDAP changelog event handlers, which can react to changes in an LDAP directory server. The event handler could be configured to run one or more assembly lines, based on the DN of the entry that was changed and the type of change (add, modify, delete, rename).

Hooks

Hooks enable the integrator to script actions to be executed under specific circumstances, or at desired points in the execution of the Assembly Line process. *JavaScript* is the preferred scripting language for ITDI because it can be used on every operating system platform for which ITDI is supported. Other scripting languages such as *Perl* can be used on some platforms.

ITDI provides a complete framework for scripting that handles all the details of invoking scripts at the appropriate point during assembly line operations and providing access to ITDI Java objects and their methods. The product's online documentation includes the complete set of APIs that users can call from scripts to access methods for controlling assembly lines, accessing data about them, calling the system utility functions provided with the product, and even executing methods on connectors.[12]

Link Criteria

Link Criteria are the attribute matching rules that locate entries in a directory for a connector to operate on when a connector is in *lookup*, *update*, or *delete* mode. When a connector is in one of these modes, ITDI automatically includes hooks in the scripting framework that enable the integrator to easily handle cases where the link criteria do not find a match in a data source or when multiple potential matches are found.

Link Criteria may be simple, such as linking two entries based on the value of a single attribute such as *employeeNumber*. More complex criteria are also easy to create by using scripts to perform data transformation if they are required for the comparison or to create complex search criteria that are specific to the data source. ITDI has built-in link comparison functions for the common "equals", "not equals", "contains", "starts with", and "ends with" comparison operations.

Work Entry

Work Entries are internal variables used to temporarily store values from directory entries. The values may be read directly from specific attributes, or may be computed by a script that performs data manipulation or transformation operations to compute a value for an attribute. Connectors are configured with *input* and *output attribute maps* that specify the mapping between the names and data types of attributes in the work object and the attribute values stored in a data source. ITDI also includes APIs that the systems integrator can call from scripts to create new attributes in the work entry and to examine and modify the values of the attributes that it contains.

---

[12] Since ITDI is 100% Java, all its public objects and methods are easily accessible from a scripting language such as JavaScript. Documentation in Javadoc form is included in the online documentation library.

Persistent store

The latest version of ITDI includes an embedded SQL database engine (IBM Cloudscape™) for Java that is useful for storing persistent metadata (not shown in the figure). Because of its lightweight, pure-Java, embeddable architecture Cloudscape is an excellent database engine for tools like ITDI because the database engine becomes part of the tool. The user never has to install or manage it and the database becomes invisible. The persistent store enables the integrator to persistently store information about data sources between runs of assembly lines. The Cloudscape persistent store can be set up so that the database is kept internal and accessible to only one ITDI configuration or put in a *network mode* that enables multiple ITDI configurations to share it.

ITDI automatically generates and uses a persistent store when a connector in iterator mode is configured to detect changes in a sequential file. When the file is processed, ITDI manages a persistent store that retains a record of the entries in the file that have been processed. This enables ITDI to recognize when new entries have been added or old entries have been deleted and efficiently return them to the connector with a flag indicating whether the entry is new, unchanged, or deleted. Connectors using this change detection mechanism have the option of indicating whether they want to receive unchanged entries, or only process adds and deletes from the file.

Older versions of ITDI include a *bTree* connector that can be used to provide a similar function. The bTree connector is retained in current versions, but the best practice is to use the more functional and Cloudscape based persistent store that relieves some constraints that affect the performance and scalability of the bTree connector.

Configuration of ITDI assembly lines

Directory Integrator provides a powerful graphical user interface (GUI) to configure the assembly lines and constituent connectors. Figure B-4 on page 699 illustrates that a configuration file can contain many different assembly lines that the system integrator can select to work with. In the figure, the assembly line named ITIMSearch has been selected. The top of the right hand panel provides a number of tabs the systems integrator may use to work with that assembly line, and the *data flow* tab has been selected.
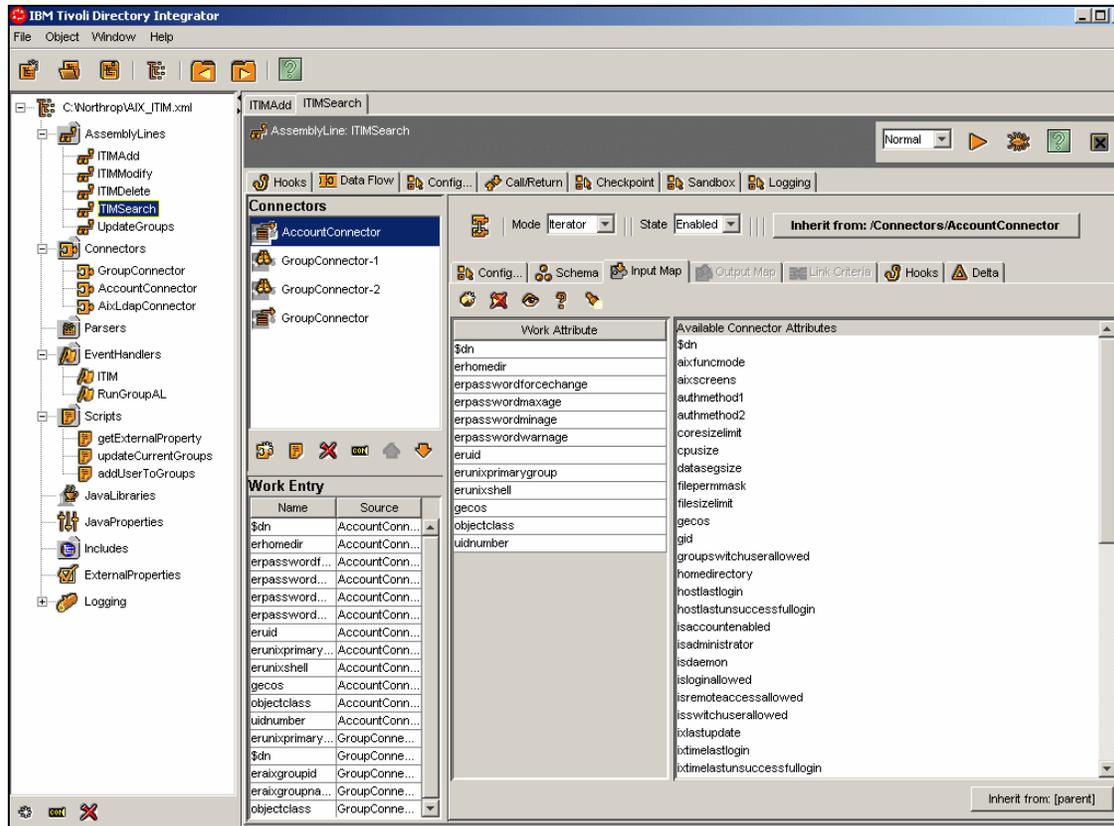
Figure B-4   ITDI Graphical Configuration Interface

The panel under the *data flow* tab is where the connectors in the *ITIMSearch* assembly line are configured and it shows the work object that flows between them. It has another row of tabs, and in the figure the input map tab has been selected. The panels on the *data flow* tab are:

► *Connectors*. This panel lists each of the connectors in the order they will be executed by the assembly line. In this example, the assembly line will run *AccountConnector* first, followed by the remaining connectors in the assembly line. Below the list of connectors are several icons to add or delete connectors from the assembly line, control the order of the connectors in the assembly line, and to add a configured connector to a connector library. Connectors in the connector library are available from the *Connectors* folder on the leftmost panel and can be selected and placed in other assembly lines. This is a powerful mechanism for re-use after a connector has been configured and tested.[13]

► *Work Entry.* This panel shows the contents of the work entry that flows
   between each of the connectors in the assembly line. Each of the attributes of
   the work entry is shown, along with the name of the connector that populates
   the work entry with the attribute value.
► When one of the connectors in the assembly line is selected, the
   configuration of that connector is displayed in the panel to the right of the
   Connectors panel. It contains another row of tabs for configuring the
   connector that has been selected.

In this example, the connector selected for configuration is in *input* mode and the
Input Map tab for it has been selected. The panel below the tab shows the
attributes that are available from the data source in the *Available Connector
Attribute* list. The integrator may drag and drop attributes from the list into the
Work Attribute list and control how they are mapped. For connectors in *update* or
add mode, ITDI will provide an *Output Map* to configure how attributes are
mapped from the work entry to the attributes available in the data source.

Notice the button labeled `Inherit from: /Connectors/AccountConnector` in the
figure that shows the attribute map is being inherited from a connector definition
in the Connectors folder. The integrator has the option of using this button to
break this inheritance if some aspect of the inherited connector configuration
needs to be handled in a specific way by selecting to inherit from another source,
or no inheritance at all. It is typical, and a best practice, for connectors to be
reused and extended in multiple assembly lines in ITDI deployments.[14]

Configuration of an ITDI Event Handler

ITDI Event Handlers are used to provide a framework to control how assembly
lines are run. This framework is particularly useful when an event, such as a
change event from an LDAP server, an HTTP message, or a JMS message is
received. The incoming event can trigger several different assembly lines to run
or other actions, depending on the content of the data received with the event.

Figure B-5 on page 702 illustrates how a typical ITDI event handler invokes an
assembly line to process an event. The event handler in the figure is the ITDS

---

[13] ITDI implements inheritance of connector configurations. There are several aspects of connector
configuration, such as the location, account names and passwords used to connect to a data source,
the source attributes retrieved or written, and hooks that implement site-specific business logic during
connector processing, etc. Internally connectors are Java objects and each of these aspects can be
inherited, extended, or replaced when a connector is reused from the library.

[14] It is possible to select a connector from the Connectors folder in the leftmost panel and perform the
configuration of a connector directly in the library. Once configured, the connectors in this folder can
be placed and used in several assembly lines. The advantage is that as the configuration is tested
and extended, changes made in the connector library can be inherited in each place the connector is
used.

changelog event handler. This event handler is configured on its configuration tab (not shown) to receive notification from an ITDS server whenever a change occurs in a part of the directory tree. The change notification event includes data about the change, such as the target DN, and the type of change.

Figure B-5 on page 702 shows the Action Map tab for the event handler that defines the processing that ITDI will perform each time a change event is received. An action map item, named *ChangeDetected* has been defined. For this item, two conditions based on the type of change received from the directory server will be evaluated to determine if the change is either an "add" or a "modify". The check box *Match Any* has been selected to indicate that if either of these conditions is true, the action items defined in the bottom panel's Condition *True* tab will be performed. In this case there are two action items. The first is to write a simple message to the console log file. The second is to run an assembly line named *PropagatePwdChg*. The *Condition False* tab may have action items that are performed if neither condition is evaluated as true. An ITDI Action Map may have multiple Action Items, each with their own set of conditions. They are evaluated in order.
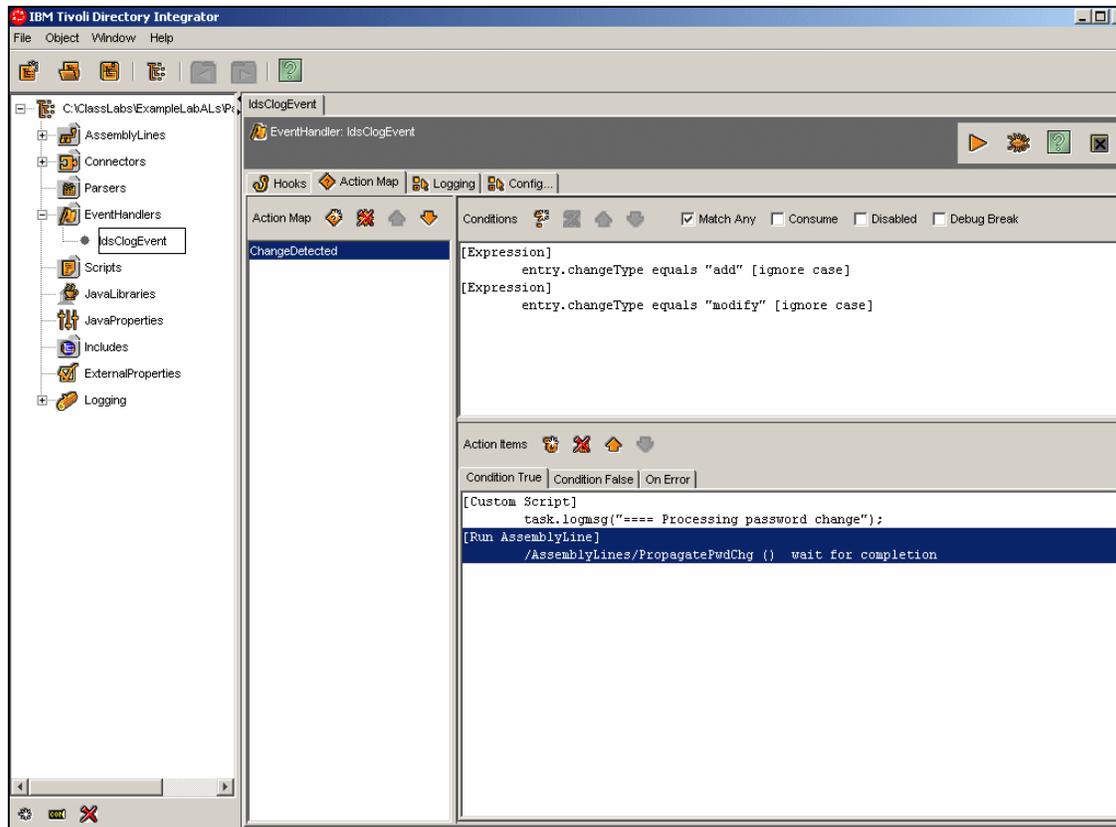
Figure B-5   ITDI Event Handler Action Map

A few Event Handlers have been pre-programmed to make things even easier. The DSMLV2 event handler shown in Figure B-6 on page 703 is an example. Here, the *Configuration* Tab contains all the information necessary to process DSML V2 events. Each of the possible DSML V2 events that can be received by the event handler are listed with a drop down list for selecting the particular assembly line that will be called to process that event. In this example, the DSML V2 *compare* and *modify* DN operations have not been associated with assembly lines, so these operations will be ignored by the event handler.
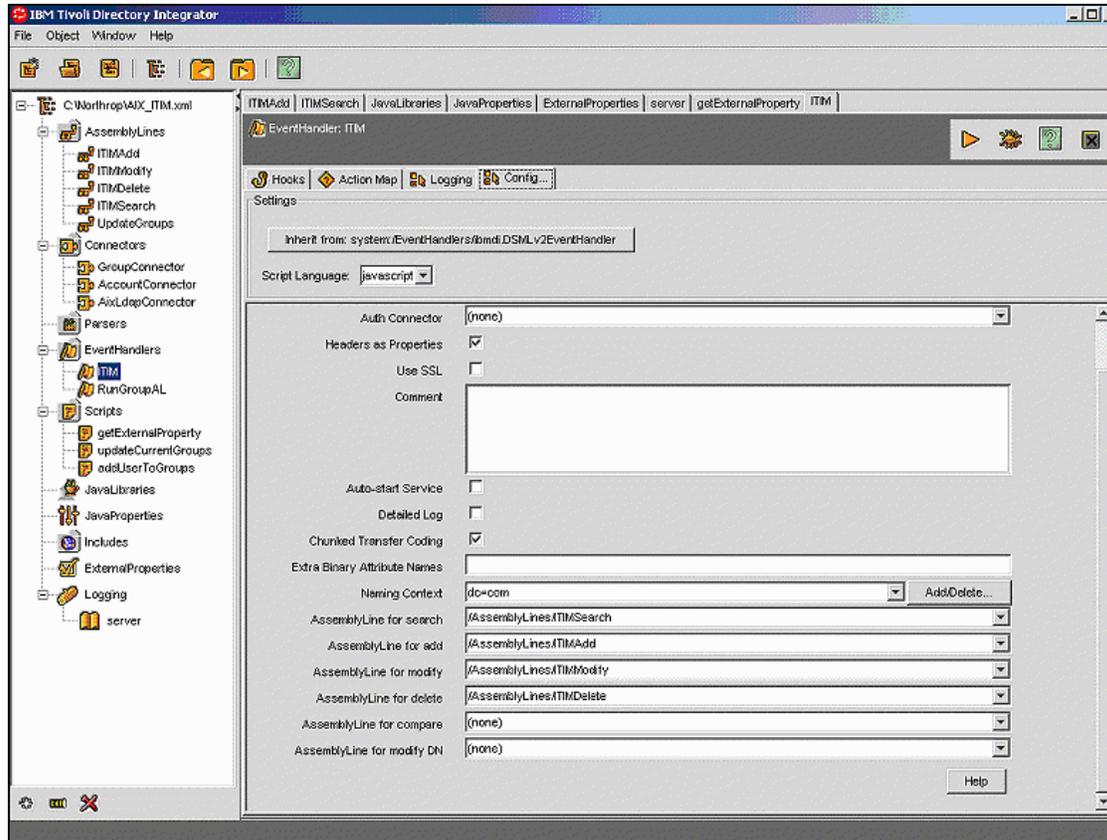
Figure B-6   ITDI DSML V2 Event Handler

ITDI solution example

With this background in directory integration technology and how it is used, we will take a look at a simplified, but typical corporate directory integration requirement and illustrate how an ITDI solution would be designed to solve it. This example has been taken from a few of the lab exercises for a training class offered by IBM for ITDI. To provide an idea of how easily integration solutions such as this can be implemented, in the training class students build all of the assembly lines described in this chapter and unit test them on the first day of class.

This solution is for an ITDI deployment to meet a mythical XYZ Company's requirement to integrate employee identity data in their HR system with Active Directory and Domino accounts. The initial objective is to load a new corporate

directory with employee data from Active Directory and Domino and keep their user data synchronized. A future project will add business partners into the corporate directory for an IBM Tivoli Access Manager for eBusiness (ITAMeB) deployment to provide authentication and access control for WebSphere applications. The IBM Tivoli Directory Server (ITDS) will be the user and group registry for ITAMeB.

XYZ Company's initial requirements are to:

- ► Load a new ITDS enterprise directory with employee data from an extract from an HR system. The HR data is merged with data from the company's existing Active Directory and Domino servers. The initial loading of the database implements a business rule configured into the ITDI assembly lines to detect and report on "dirty" data that is discovered in the legacy Active Directory and Domino servers.
- ► Keep the enterprise directory, Domino server, and Active Directory synchronized with a daily extract of a simulated HR system.
  - When a new user is added to the HR system, an Active Directory account is created. A Domino account can also be created with an email database and a Notes ID file for use with the Notes thick client. New Notes users are also provisioned with an HTTP password so they can access email using the Notes browser interface.
  - When a user's status is changed from "Active" to "Inactive" in the simulated HR system, the user's Active Directory account is disabled. When a user's status is changed from "Inactive" to "Active" their existing Active Directory account is enabled.
- ► No significant changes will be made to XYZ Company's current business processes for maintaining identity data in the IT infrastructure.   Current XYZ Company business processes may be streamlined, but administrative interfaces for managing user identities will not be modified or extended in this design.
- ► All XYZ Company employees will have an entry in the new corporate LDAP directory. They may also be given Active Directory and Domino accounts or added to static groups in the LDAP directory, based on their job code.
- ► When an XYZ Company HR administrator creates a new employee record in the HR system, the employee is assigned a unique employee User ID. The employee User ID will be the ID the employee authenticates with in the LDAP directory, Domino and Active Directory.   Employees are never deleted from the HR system. Their status is changed from *active* to *inactive* on their termination date.
- ► User accounts will also be created in Active Directory, ITAMeB and Domino. Which accounts are created is based on the employee's job code.
  - Users with a Job Code greater than 9 are knowledge workers. Knowledge workers always require Active Directory, ITAMeB, and Domino accounts.

– Users with a Job of 9 or less are production workers. Production workers do not need Domino accounts for email or to access other Domino applications. Production workers require an Active Directory account to access time and attendance and other shop floor applications that run on workstations or servers in the XYZ Company Windows domain. After the future ITAMeB deployment, they will require an ITAMeB account to access browser based HR applications from home, so they must also be in the LDAP directory.

► The XYZ Company Active Directory and Domino servers already contain entries for all current employees.

ITDI solution design

The ITDS LDAP directory is new at XYZ Company. It must be initially populated from the set of currently active employees in the HR system. There may be some "dirty" data in Domino and Active Directory. For example, employees that have job codes that require Domino accounts that do not have them or employees who do not have Active Directory accounts.

After the initial population of the LDAP directory, XYZ Company will use IDI to add new employees to the LDAP directory when they are added to the HR system. When employee information is changed in the HR system IDI will keep the directory, Domino, and Active Directory data synchronized.   IDI assembly lines will support the following events.

► New Users. When new employees are added to HR, they must be added to the LDAP directory and assigned a Windows user ID and password.   If the employee's job code is greater than 9, they are a knowledge worker, and a Domino account will be created.

► Account Revocation. When an employee leaves XYZ Company, their employee status is changed from an Active to Inactive in the HR system. ITDI must deactivate their Windows accounts.

► Employee Status Changes. When a user's job code changes, an ITDI assembly line must evaluate whether the employee needs a new Domino account created. If so, one will be created.

HR System Extract

The authoritative source for most identity information is an extract from the HR system. This extract is generated by a utility provided that produces a "comma delimited" flat file and contains the following data for each employee. The first

record of the flat file is a header record containing the names of the fields in the following records. There is one record for each employee that contains:

- ► User ID
- ► Street address
- ► State
- ► City
- ► Zip code
- ► First name
- ► MI (middle initials)
- ► Last name
- ► Office fax number
- ► Full name
- ► Job code
- ► Employee number
- ► Employee status
- ► Dept number
- ► Telephone number
- ► Organizational title

Active Directory

Active Directory users are stored in the *cn=Users,dc=xyz,dc=com* container. The RDN is the *cn* attribute. XYZ Company has been using the *full name* as the value of this attribute but this has caused problems. They have decided to use the unique UID created by the HR system for new users. Active Directory will be the authoritative source for UserPrincipalName. This attribute in Active Directory will be the User ID@XYZ.com. For example, for the user with the ID *jdoe*, it will be jdoe@xyz.com. It will be mapped to the *description* attribute in the IDS LDAP directory.

Another important attribute in Active Directory is userAccountControl. This Active Directory attribute controls the active or inactive status of a user account, among other things. If the second low order bit of this attribute is set on, the account is inactive.[15]

Domino

For the initial phase of the project the user's email address (the *mail* attribute) will be retrieved from the Domino server to populate the LDAP directory. When IDI creates new users in Domino, the basic attributes required by the Domino User connector to register a new user will be given values.

---

[15] See Microsoft documentation for a complete description of the userAccountControl attribute.

XYZ Company ITDS Directory Information Tree

The XYZ Company LDAP directory tree structure is illustrated in Figure B-7. The suffix of the directory is *dc=XYZ Company,dc=com*. The Directory Information Tree (DIT) is organized into a hierarchy of containers, or organizational units (*ou*'s).
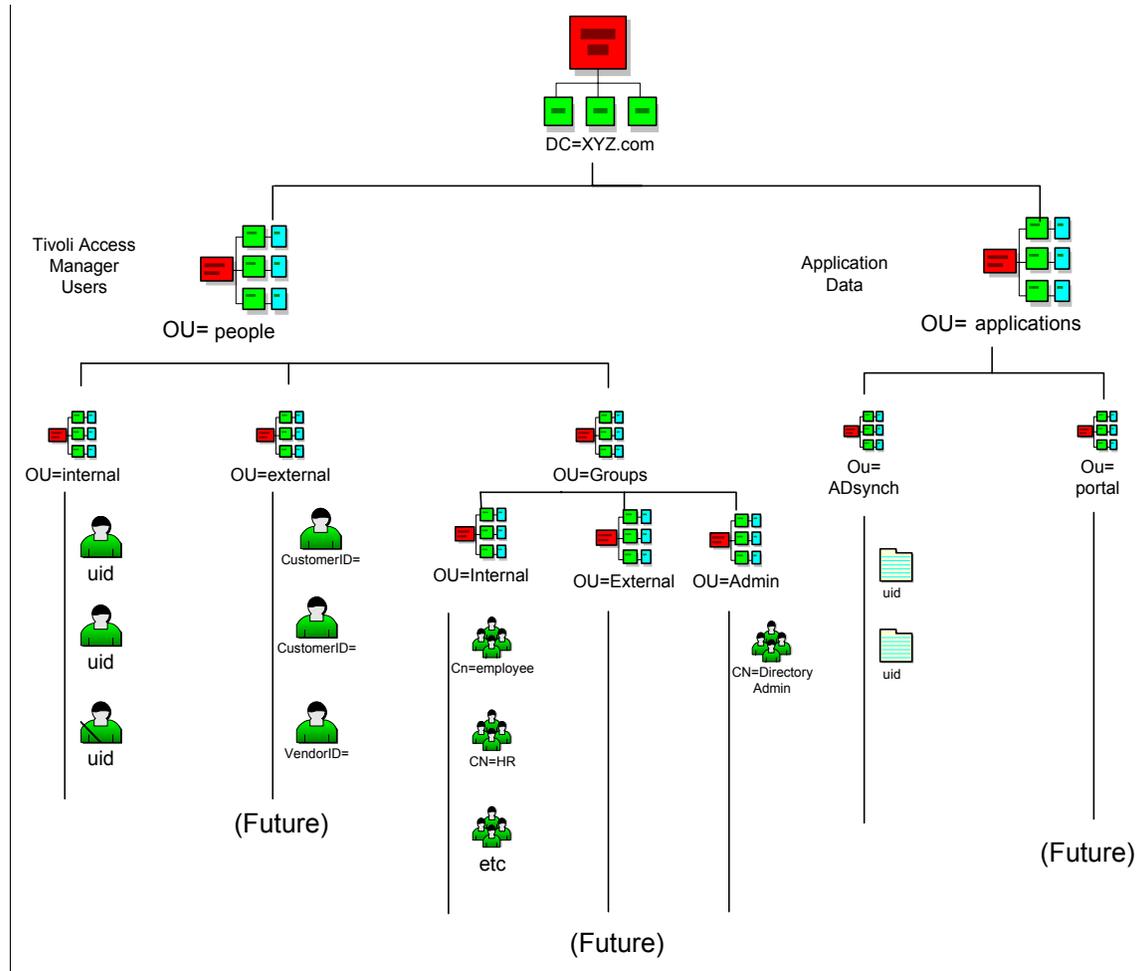
Figure B-7   XYZ Company Directory Information Tree (DIT)

User and group containers

The *ou=Internal* container will store entries for employees. under *ou=people, dc=XYZ, dc=com*. The subtree is flat to avoid embedding organizational structure into the Distinguished Names in the directory. The Relative

Distinguished Name (RDN) for entries in this container will be the *uid* attribute that contains the user's Windows ID.

The *ou=External* container is reserved for future use. Only internal users (employees and contractors) will be stored in the directory in this phase of the XYZ Company directory integration solution. In the future, external users (customers and vendors) can be added under the container *ou=External*.

The *ou=Groups* container will store groups of users under the subtree *ou=people, dc=XYZ,dc=com*. The group subtree will be subdivided into containers for groups of internal users, external users, and administrative users. This enables the directory administrator to keep groups of administrators, internal users, and external users.

The RDN of the group entries in the directory is the *cn* attribute (the *common name* of the group). The three group containers are:

- ▶ *ou=internal,ou=groups,ou=people,dc=xyz.com*. ITAMeB will be deployed in the future to control access to resources based on the groups to which users belong. This container will contain the groups used by ITAMeB to control access to the resources it protects. In the figure, for example, the members of the group *cn=employee*, *ou=internal*, *ou=groups*, *ou=people*, *dc=xyz, dc=com* will be given access to the XYZ employee portal by ITAMeB.

- ▶ *ou=external,ou=groups,ou=people,dc=xyz.com*. This subtree is reserved. In the future groups containing external users may be defined in this subtree.

- ▶ *ou=admin,ou=groups,ou=people,dc=xyz.com*. This subtree contains administrative groups whose members are users given administrative rights for ITAMeB, applications, and the directory server.

Application container

Directory entries in the *ou=Applications* container will be used by directory enabled XYZ Company infrastructure components and applications.

- ▶ *ou=ADsynch, ou=Applications,dc=XYZ, dc=com*. In the future, after ITAMeB is installed, the ITDI Active Directory password interceptor will be installed on the Active Directory Domain Controller to synchronize passwords in the ITDS directory when users change their passwords in Active Directory. The ITDI interceptor will securely store the public key encrypted new password in this ITDS container. An ITDI assembly line will propagate new or changed values for *password* to the entry with the same *uid* in the *ou=Internal, ou=People, dc=XYZ,dc=com* container.

- ▶ *ou=Portal, ou=Applications,dc=xyz.com*. This container will be used in the future by the WebSphere Portal Server and portal applications.

LDAP Schema

In this design the schema and indexing information in the tables in this section are the default configuration of the IBM Directory Server "out of the box" after installation. The following attributes will be populated in the directory:

- ► *cn* (Common Name) - This *multivalued* attribute will contain two values, the user's full name in HR and the user's last name to facilitate LDAP directory searches based on the last name.

- ► *businessCategory* - This attribute will contain the employee's job code from HR.

- ► *departmentNumber* - From HR.

- ► description - This attribute will contain the *userPrincipalName* in Active Directory.

- ► *displayName* - The user's full name from HR.

- ► *employeeNumber* - From HR.

- ► *employeeType* - Mapped to employee status in HR.

- ► *facsimileTelephoneNumber* - From office fax number in HR.

- ► *givenName* - From first name in HR.

- ► *initials* - From MI in HR.

- ► *l* (location) - From the city attribute in HR.

- ► *mail* - The email address from Domino.

- ► *postalCode* - From zip code in HR.

- ► *sn* - From last name from HR.

- ► *st* - From state in HR.

- ► *street* - From street address in HR.

- ► *telephoneNumber* - From telephone number in HR.

- ► *title* - From organizational title in HR.

- ► *uid* - From user ID in HR.

Groups will be created in the *ou=Groups,ou=People,dc=XYZ,dc=com* container with the *accessGroup* objectclass.   The future Access Manager deployment will use groups in ACLs that control user access to the resources it protects.

Solution components

The components of the XYZ Company Directory and future ITAMeB deployment are illustrated in Figure B-8.
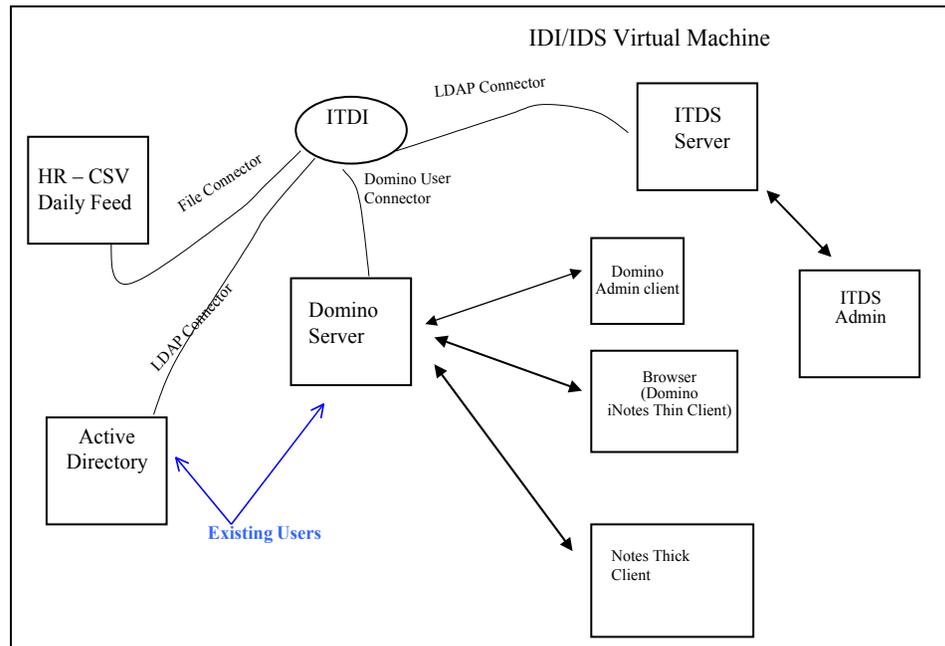
Figure B-8   Solution architecture

Initial population of the LDAP Directory

This section describes how the LDAP directory will be initially loaded with entries for current XYZ Company employees. Once populated, the LDAP directory will be maintained by the procedures and ITDI assembly lines described in the next section.

The initial entries loaded into the directory will contain attributes that are populated from the XYZ Company HR System. The extract utility provided with the HR system will be used to export user data into an XML file. An extract is generated that contains the HR information for each current employee. The data flow through the assembly line for initial population of the LDAP directory is shown in Figure B-9 on page 711.[16]

---

[16] In this and other data flow diagrams in this document, components, such as the HR Extract Utility in this figure that are external to this system are shown as rectangles.
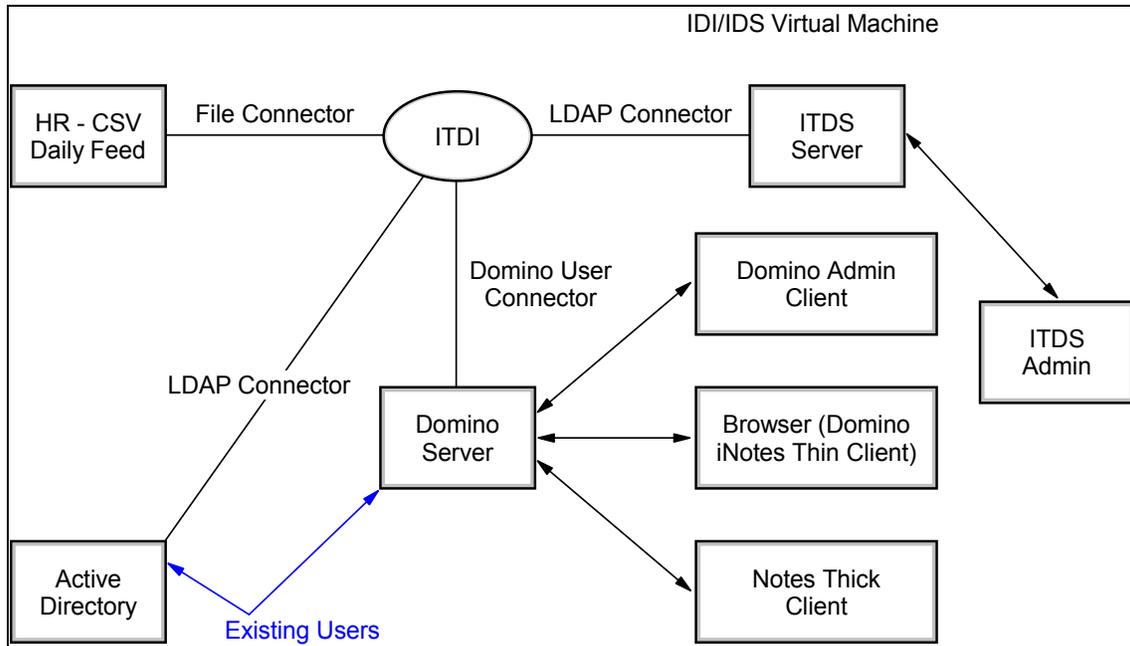
Figure B-9   Initial Directory population data flow

This assembly line contains the following connectors:

- ► File System Connector - This connector will be configured in iterator mode to read the XML data exported from the HR System. Into the ITDI work object. It will pass the entry to the next connector.

- ► LDAP Connector (Domino) - This connector will be configured in Lookup mode. It will be configured to look up the employee's email address in Domino and add it to the work entry. Only employees with a job code of 10 or higher should have an email address. If "dirty data" is found, a passive state connector[17] will be called to output an error message.

- ► LDAP Connector (Active Directory) - This connector will be configured in Lookup mode. It will lookup the user's userPrincipalName in Active Directory. If no Active Directory account information is found for an active employee, a passive state connector will be called to output an error message. An SSL connection will be configured to Active Directory in preparation for the requirement to add active directory users and set their passwords.[18]

---

[17] A connector can be placed in an enabled, disabled, or passive state (see the drop down list for these states in Figure B-4 on page 699). In passive state, the ITDI framework does not automatically call the connector. The connector is typically called from a hook script when needed, such as in this case when an error condition occurs.

[18] Active Directory requires an SSL session be used when setting user passwords.

- ▶ LDAP Connector (ITDS *ou=internal, ou=People* container) - This connector will be configured in Update mode. It will lookup the entry in the IDS directory for the employee with the data from the HR system[19]. If found, the connector will update the attributes in its attribute map with the values from the work entry. If not found, the connector will add the entry.

  The first time the assembly line runs, the entry will not be found, since the LDAP directory is empty, and the connector will create a new entry from the HR data, email address from Domino, and the userPrincipalName from Active Directory. If the assembly line is run multiple times, the user entry will be found if it was created by a previous run. The entry will be updated if the current HR data differs from the data in the directory.

- ▶ File System Connector - This connector will be configured in passive state with an XML parser. It will be called by other connectors to write errors to an error log.

User Management

This section describes the IDI data flows that will keep the LDAP directory, and Active Directory synchronized. These data flows are triggered by new or changed entries in the HR system.   The processes supporting these events are shown in the data flow diagram in Figure B-10 on page 713.

---

[19] It could also be configured in add mode, but existing entries need to be handled if the assembly line is run multiple times during testing.
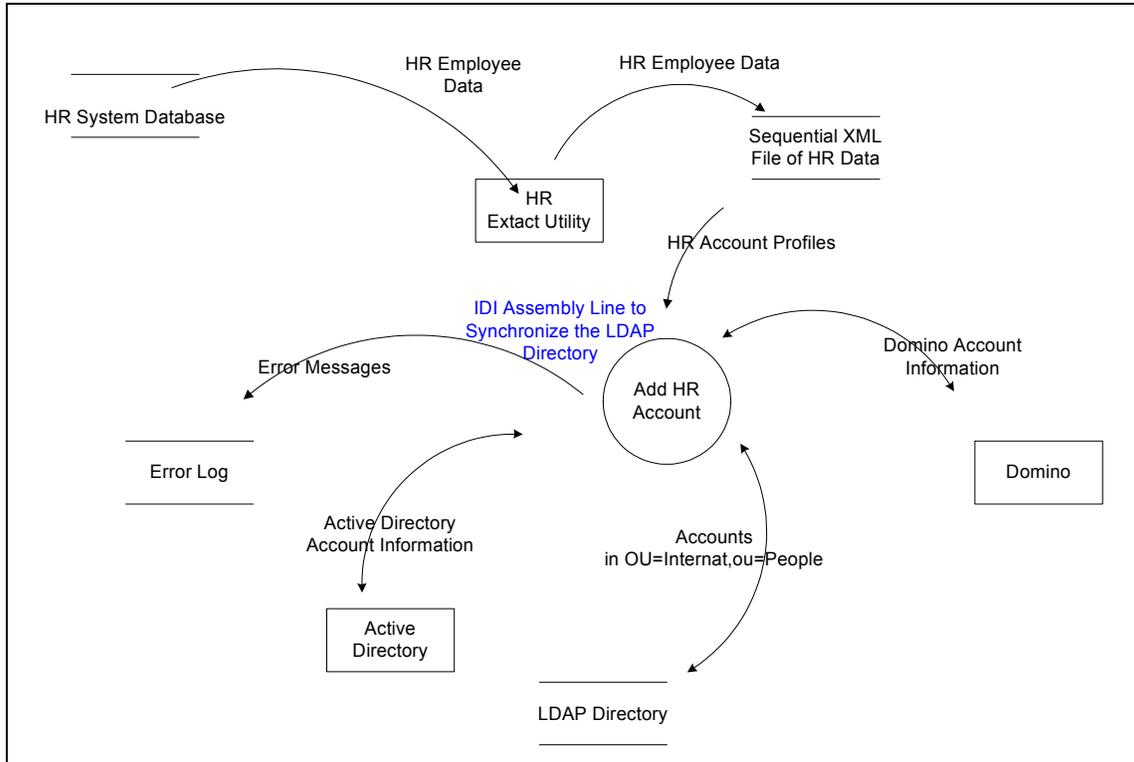
Figure B-10 User management data flow

This assembly line contains the following connectors:

► Flat file Connector - This connector will be configured in iterator mode to read the XML data exported from the HR System.

► Domino User Connector - This connector will be configured in Update Mode. If this is a potential new Domino user, and job code requires an email account, it will create a new Domino account for the user. Otherwise, the connector will update the entry if the HR Data has changed.

► LDAP Connector (Active Directory) - This connector will be configured in update mode. It will lookup the user's userPrincipalName in Active Directory. If no Active Directory account information is found for an active employee a new account will be created. The employee status attribute from the HR file will be mapped to the Active Directory userAccountControl attribute to enable or disable the Active Directory account based on their current status.

► LDAP Connector (ITDS *ou=internal, ou=People* container) - This connector will be configured in Update mode. It will lookup the entry in the ITDS directory for the employee with the data from the HR system.

Summary

In this chapter we have reviewed the reasons why it is usually critical to address directory integration requirements in any major directory deployment project. We reviewed several methodologies for integrating directories with heterogeneous legacy data stores and focused on metadirectories as currently providing the most robust and stable technology. Metadirectories were compared to the complementary technologies of provisioning systems and virtual directories. Finally we provided an overview of a particular metadirectory, the IBM Tivoli Directory Integrator and illustrated a solution design for deploying it to solve a set of relatively simple but realistic and common integration requirements.
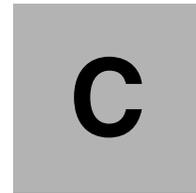
Moving RACF users to TBDM

This appendix provides a sample program that can move RACF users into the TDBM space as discussed in "Moving RACF users to the TDBM space" on page 189.

Sample programs to move RACF users to TBDM

There are three pieces that are needed to convert RACF users to TDBM users. They are used in a compound command line that is invoked by issuing the following command:

```
search1 | xargs -i search2 {} | racf2person > output.ldif
```

The three pieces that are used in this example are **seach1**, **search2**, and **racf2person**. The **search1** command is a **ldapsearch** command that will return all the RACF distinguished names and is shown in Example C-1.

Example: C-1   search1 command

```
ldapsearch -h 1.1.ibm.com:389 \
  -D racfid=root,profiletype=user,sysplex=testplex -w XXXXXX \
  -s one -b profiletype=user,sysplex=testplex "(racfid=A*)"
```

The **search2** piece is a Perl program and will search for the RACF user attributes and is shown in Example C-2.

Example: C-2   search2 Perl program

```
# do a search
name=$1

ldapsearch -h 1.1.ibm.com:389 \
  -D racfid=root,profiletype=user,sysplex=testplex -w XXXXXX \
  -L -s base -b $name "(objectclass=*)"

echo ""
```

The **racf2person** piece is also a Perl program that will convert the RACF user information (distinguished name from **search1** and user attributes from **search2**) into the LDIF format. The *racf2person* program is shown in Example C-3.

Example: C-3   racf2person Perl program

```
#!/usr/bin/perl
eval "exec perl -S $0 $*"
    if $running_under_some_shell;
#
# <COPYRIGHT>
# Copyright (c) 2003, International Business Machines Corporation
# and others. All Rights Reserved.
#
# For the full copyright information for this source code, refer to
# the COPYRIGHT file in the root directory of the source code tree.
# </COPYRIGHT>
```

```
#

#
# \file
#
# \version 1.0
#
# This is the Makefile for the src/doc directory.
#
# <PRE>
# Change History
# Date          Name            Description
# ----------------------------------------------------------------
# 2003/10/09  Tim Hahn          Created this file
# </PRE>
#

# handle input parms
if ( $#ARGV+1 < 2 ) {
   do Usage();
   exit(-1);
}
else {
   for ($i=0; $i<$#ARGV+1; $i++) {
      if ($ARGV[$i] eq "-b") {
         $suffixDN = $ARGV[$i+1];
         $i++;
      }
      else {
         do Usage();
         exit(-1);
      }
   }

   @ARGV=(); # clear the input parms
}

do ResetParms();

while (<>) {
   /^dn: .*/ && ($outputReady==1) && do { # signifies the start of another
user, print what we have

      if ($commonName eq "") { # if not set, use the uid value
         $commonName = $uid;
         $surName = $uid;
      }

      do PrintPerson();
```

```
            do ResetParms();
            };

    /^racfid: .*/ && do { # take the racfid and use it as uid and
ibm-nativeID
        $uid = $_;
        $uid =~ s/\n//; # remove the newline
        $uid =~ s/^racfid: //;
        $nativeID = $uid;

        $outputReady = 1;
        };

    /^racfprogrammername: .*/ && do { # take the racfprogrammername and use
it for cn and sn
        $commonName = $_;
        $commonName =~ s/\n//; # remove the newline
        $commonName =~ s/racfprogrammername: //;

        $surName = $commonName;
        $surName =~ s/^[^ ]* //; # remove the first name
        };
}

if ( $outputReady == 1 ) {
    if ($commonName == "") { # if not set, use the uid value
        $commonName = $uid;
        $surName = $uid;
    }

    do PrintPerson();
}


sub Usage {
    print("Usage: racf2Person < -b <suffixDN> >\n");
}

sub PrintPerson {
    print ("dn: cn=", $commonName, ",", $suffixDN, "\n");
    print ("objectclass: person\n");
    print ("objectclass: inetorgPerson\n");
    print ("objectclass: ibm-nativeAuthentication\n");
    print ("cn: ", $commonName, "\n");
    print ("sn: ", $surName, "\n");
    print ("uid: ", $uid, "\n");
    print ("ibm-nativeID: ", $nativeID, "\n");
    print ("\n");
```

```
}

sub ResetParms {
    $outputReady=0;

    $uid="";
    $nativeID="";
    $commonName="";
    $surName="";
}
```

D

Schema changes that are not allowed

This appendix provides a list of schema changes that are not allowed.

Operational attributes

These are the operational attributes that cannot be modified:

- ► aclEntry
- ► aclPropagate
- ► aclSource
- ► aliasedObjectName, aliasedentryName
- ► createTimestamp
- ► creatorsName
- ► entryOwner
- ► hasSubordinates
- ► ibm-allGroups
- ► ibm-allMembers
- ► ibm-capabilitiessubentry
- ► ibm-effectiveAcl
- ► ibm-entryChecksum
- ► ibm-entryChecksumOp
- ► ibm-entryUuid
- ► ibm-filterAclEntry
- ► ibm-filterAclInherit
- ► ibm-replicationChangeLDIF
- ► ibm-replicationIsQuiesced
- ► ibm-replicationLastActivationTime
- ► ibm-replicationLastChangeId
- ► ibm-replicationLastFinishTime
- ► ibm-replicationLastGlobalChangeId
- ► ibm-replicationLastResult
- ► ibm-replicationLastResultAdditional
- ► ibm-replicationNextTime
- ► ibm-replicationPendingChangeCount
- ► ibm-replicationPendingChanges
- ► ibm-replicationState
- ► ibm-replicationThisServerIsMaster
- ► modifiersName
- ► modifyTimestamp
- ► ownerPropagate
- ► ownerSource
- ► pwdAccountLockedTime
- ► pwdChangedTime
- ► pwdExpirationWarned
- ► pwdFailureTime
- ► pwdGraceUseTime
- ► pwdHistory
- ► pwdReset

- ► subschemaSubentry
- ► subtreeSpecification

Restricted attributes

These are the restricted attributes that cannot be modified:

- ► aclEntry
- ► aclPropagate
- ► entryOwner
- ► ibm-filterAclEntry
- ► ibm-filterAclInherit
- ► ownerPropagate

Root DSE attributes

These are the Root DSE attributes that cannot be modified:

- ► altServer
- ► ibm-effectiveReplicationModel
- ► ibm-enabledCapabilities
- ► ibm-serverId
- ► ibm-supportedCapabilities
- ► ibm-supportedReplicationModels
- ► namingContexts

Schema definition attributes

These are the Schema Definition attributes that cannot be modified:

- ► attributeTypes
- ► ditContentRules
- ► ditStructureRules
- ► IBMAttributeTypes
- ► ldapSyntaxes
- ► matchingRules
- ► matchingRuleUse
- ► nameForms
- ► objectClasses
- ► supportedExtension
- ► supportedLDAPVersion
- ► supportedSASLMechanisms

Configuration attributes

These are the Configuration attributes that cannot be modified:

► ibm-audit
► ibm-auditAdd
► ibm-auditBind
► ibm-auditDelete
► ibm-auditExtOpEvent
► ibm-auditFailedOpOnly
► ibm-auditLog
► ibm-auditModify
► ibm-auditModifyDN
► ibm-auditSearch
► ibm-auditUnbind
► ibm-slapdAclCache
► ibm-slapdAclCacheSize
► ibm-slapdAdminDN
► ibm-slapdAdminPW
► ibm-slapdAuthIntegration
► ibm-slapdCLIErrors
► ibm-slapdDB2CP
► ibm-slapdDBAlias
► ibm-slapdDbConnections
► ibm-slapdDbInstance
► ibm-slapdDbLocation
► ibm-slapdDbName
► ibm-slapdDbUserID
► ibm-slapdDbUserPW
► ibm-slapdDerefAliases
► ibm-slapdDN
► ibm-slapdsupportedCapabilities
► ibm-slapdEnableEventNotification
► ibm-slapdEntryCacheSize
► ibm-slapdErrorLog
► ibm-slapdFilterCacheBypassLimit
► ibm-slapdFilterCacheSize
► ibm-slapdIdleTimeOut
► ibm-slapdIncludeSchema
► ibm-slapdIpAddress
► ibm-slapdKrbAdminDN
► ibm-slapdKrbEnable
► ibm-slapdKrbIdentityMap
► ibm-slapdKrbKeyTab
► ibm-slapdKrbRealm

- ► ibm-slapdLdapCrlHost
- ► ibm-slapdLdapCrlPassword
- ► ibm-slapdLdapCrlPort
- ► ibm-slapdLdapCrlUser
- ► ibm-slapdMasterDN
- ► ibm-slapdMasterPW
- ► ibm-slapdMasterReferral
- ► ibm-slapdMaxEventsPerConnection
- ► ibm-slapdMaxEventsTotal
- ► ibm-slapdMaxNumOfTransactions
- ► ibm-slapdMaxOpPerTransaction
- ► ibm-slapdMaxTimeLimitOfTransactions
- ► ibm-slapdMigrationInfo
- ► ibm-slapdPagedResAllowNonAdmin
- ► ibm-slapdPagedResLmt
- ► ibm-slapdPageSizeLmt
- ► ibm-slapdPlugin
- ► ibm-slapdPort
- ► ibm-slapdslapdPwEncryption
- ► ibm-slapdReadOnly
- ► ibm-slapdReferral
- ► ibm-slapdSchemaAdditions
- ► ibm-slapdSchemaCheck
- ► ibm-slapdSecurePort
- ► ibm-slapdSecurity
- ► ibm-slapdSetenv
- ► ibm-slapdSizeLimit
- ► ibm-slapdSortKeyLimit
- ► ibm-slapdSortSrchAllowNonAdmin
- ► ibm-slapdSslAuth
- ► ibm-slapdSslCertificate
- ► ibm-slapdSslCipherSpec
- ► ibm-slapSslCipherSpecs
- ► ibm-slapdSslKeyDatabase
- ► ibm-slapdSslKeyDatabasePW
- ► ibm-slapdSslKeyRingFile
- ► ibm-slapdSslKeyRingFilePW
- ► ibm-slapdSuffix
- ► ibm-slapdSupportedWebAdmVersion
- ► ibm-slapdSysLogLevel
- ► ibm-slapdTimeLimit
- ► ibm-slapdTraceEnabled
- ► ibm-slapdTraceMessageLevel
- ► ibm-slapdTraceMessageLog
- ► ibm-slapdTransactionEnable

- ibm-slapdUseProcessIdPW
- ibm-slapdVersion
- replicaBindDN
- replicaBindMethod
- replicaCredentials, replicaBindCredentials
- replicaHost
- replicaPort
- replicaUpdateTimeInterval
- replicaUseSSL

User Application attributes

These are the User Application attributes that cannot be modified:

- businessCategory
- cn, commonName
- changeNumber
- changes
- changeTime
- changeType
- deleteOldRdn
- description
- dn, distinguishedName
- member
- name
- newSuperior
- o, organizationName, organization
- objectClass
- ou, organizationalUnit, organizationalUnitName
- owner
- ref
- seeAlso
- targetDN

Abbreviations and acronyms

ACI	Access Control Interface	**DARPA**	Defense Advanced Research Projects Agency
ACL	Access Control List	**DAS**	Directory Assistance Service
ADSI	Active Directory Service Interface	**DCD**	Document Content Description
AIX	Advanced Interactive Executive	**DEN**	Directory-Enabled Networks Initiative
ANSI	American National Standards Institute	**DES**	Data Encryption Service
API	Application Programming Interface	**DIT**	Directory Information Tree
ASCII	American National Standard Code for Information Interchange	**DMTF**	Desktop Management Task Force
BER	Basic Encoding Rules	**DN**	Distinguished Name
BNF	Backus Naur Form	**DNS**	Domain Name Service
CA	Certificate Authority	**DOS**	Denial Of Service
CCITT	International Consultative Committee on Telephony and Telegraphy	**DSML**	Directory Services Markup Language
		EH	Encrypted Header
CGI	Computer Graphics Interface	**FIPS**	Federal Information Processing Standard
CIM	Common Information Model	**FTP**	File Transfer Protocol
CLI	Command Line Interface	**GSKIT**	IBM Global Security Toolkit
CN	Common Name	**GUI**	Graphical User Interface
CPAN	Comprehensive Perl Archive Network	**HTML**	Hyper Text Markup Language
DAML	Directory Access Markup Language	**HTTP**	Hyper Text Transfer Protocol
DAP	Directory Access Protocol	**HTTPS**	Hyper Text Transfer Protocol over SSL

727

IAB	Internet architecture Board	**JMS**	Java Message Service
IANA	Internet Assigned Numbers Authority	**JNDI**	Java Naming and Directory Interface
IBM	International Business Machines Corporation	**JPEG**	Joint Photographics Expert Group
IETF	Internet Engineering Task Force	**JRE**	Java Runtime Environment
IMAP	Internet Mail Access Protocol	**JSP**	Java Server Page
IP	Internet Protocol	**KDC**	Key Distribution Center
ISBN	International Standard Book Number	**LDAP**	Lightweight Directory Access Protocol
ISI	Information Sciences Institute	**LDIF**	LDAP Data Interchange Format
ISO	International Standards Organization	**LIPS**	Lightweight Internet Person Schema
ITDI	IBM Tivoli Directory Integrator	**MAC**	Machine Address Code
ITDS	IBM Tivoli Directory Server	**MIME**	Multipurpose Internet Mail Extensions
ITSO	International Technical Support Organization	**OID**	Object Identifier
		OS	Operating System
ITU	International Telecommunications Union	**OSI**	Open Systems Interconnect
ITU-T	International Telecommunications Union - Telecommunication Standardization Sector	**PDF**	Portable Document Format
		PID	Process Identifier
		RACF	Resource Access Control Facility
JAR	Jave Archive	**RAM**	Randon Access Memory
JDBC	Java Database Connectivity	**RDBMS**	Relational Database Management System
JDK	Java Development Kit	**RDN**	Relative Distinguished Name
JLDAP	Java LDAP	**RFC**	Request For Comments
		RPC	Remote Procedure Call

RSA	Rivest-Shamir-Adleman algorithm	
SASL	Simple Authentication and Security Layer	
SDK	Software Development Kit	
SHA	Secure Hash Algorithm	
SMP	Shared Multi-Processor	
SMTP	Simple Mail Transfer Protocol	
SNMP	Simple Network Management Protocol	
SOAP	Simple Object Access Protocol	
SPI	Service Provider Interface	
SQL	Stuctured Query Language	
SSL	Secure Sockets Layer	
TCP	Transmission Control Protocol	
TGT	Ticket Granting Ticket	
TLS	Transport Layer Security	
TTY	Teletypewriter	
UID	User Identification	
URI	Universal Resource Identifier	
URL	Universal Resource Locator	
UUID	Universal Unique Identifier	
VM	Virtual Machine	
XML	eXtensible Markup Language	

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see "How to get IBM Redbooks" on page 733. Note that some of the documents referenced here may be available in softcopy only.

► *Understanding LDAP,* SG24-4986

► *LDAP Implementation Cookbook*, SG24-5110

► *Using LDAP for Directory Integration*, SG24-6163

Online resources

These Web sites and URLs are also relevant as further information sources:

► ADSI information:

http://www.microsoft.com/windows2000/techinfo/howitworks/activedirectory/adsilinks.asp

► Apache Directory Project:

http://incubator.apache.org/directory/subprojects/eve/index.html

► ASN.1 frequently asked questions:

http://asn1.elibel.tm.fr/oid/faq.htm

► Directory Interoperability Forum:

http://www.opengroup.org/dif/

► DirectoryMark:

http://www.mindcraft.com/directorymark/index.html

► DSML information:

http://www.dsmltools.org

► IBM DB2 Universal Database Product Documentation:

http://www.ibm.com/software/data/db2/library/

- IBM Tivoli Directory Server information:

  http://www-306.ibm.com/software/tivoli/products/directory-server/

- IBM Tivoli Directory Server Product Documentation:

  http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html

- IBM Tivoli Directory Server Schema information:

  http://publib.boulder.ibm.com/tividd/td/IBMDS/IDSschema52/en_US/HTML/schema.html

- International Standards Organization:

  http://www.iso.ch/

- Internet Assigned Numbers Authority:

  http://www.iana.org/cgi-bin/enterprise.pl

- Internet Engineering Task Force (IETF):

  http://www.ietf.org/

- ITU:

  http://www.itu.ch/

- Java LDAP browser:

  http://www.iit.edu/~gawojar/ldap/index.html

- Java SDK 1.3.1:

  http://www.alphaworks.ibm.com/aw.nsf/download/xml4j

- JNDI information:

  http://java.sun.com/products/jndi/

- JXplorer:

  http://pegacat.com/jxplorer/

- LDAPZone:

  http://www.ldapzone.com/

- Mozilla:

  http://www.mozilla.org/directory/

- NET::LDAP:

  http://search.cpan.org/~gbarr/perl-ldap-0.31/

- OpenLDAP:

  http://www.openldap.org

- Request for comments:

  http://www.ietf.org/rfc/rfc.html

- ► SOAP 2.3:

  http://xml.apache.org/dist/soap/version-2.3

- ► Understanding X.500 - The Directory:

  http://www.isi.salford.ac.uk/staff/dwc/X500.htm

- ► Unicode Character Encoding:

  http://www.unicode.org/

- ► University of Michigan LDAP Mailing List Archives:

  http://listserver.itd.umich.edu/cgi-bin/lyris.pl?visit=ldap

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads:

ibm.com/support

IBM Global Services:

ibm.com/services

Index

IBM

Redbooks

Understanding LDAP
Design and Implementation

Understanding LDAP Design and Implementation

IBM®

Redbooks

LDAP concepts and architecture

Designing and maintaining LDAP

Step-by-step approach for directory implementation

The implementation and exploitation of centralized, corporate-wide directories are among the top priority projects in most organizations. The need for a centralized directory emerges as organizations realize the overhead and cost involved in managing the many distributed micro and macro directories introduced in the past decade with decentralized client/server applications and network operating systems.

Directories are key for successful IT operation and e-business application deployments in medium and large environments. IBM understands this requirement and supports it by providing directory implementations based on industry standards at no additional cost on all its major platforms and even important non-IBM platforms. The IBM Directory Server implements the Lightweight Directory Access Protocol (LDAP) standard that has emerged quickly in the past years as a result of the demand for such a standard.

This IBM Redbook will help you create a foundation of LDAP skills, as well as install and configure the IBM Directory Server. It is targeted at security architects and specialists who need to know the concepts and the detailed instructions for a successful LDAP implementation.