Using LDAP for Directory Integration

A Look at IBM SecureWay Directory, Active Directory and Domino

- LDAP integration guidelines for systems administrators
- Referrals and schema extensions
- Examples to help you integrate directories

Harri Levanen
Bernard Freund
Hani Mansi

Redbooks

ibm.com/redbooks

SG24-6163-00

International Technical Support Organization

Using LDAP for Directory Integration
A Look at IBM SecureWay Directory, Active Directory and Domino

December 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix K, “Special notices” on page 213.

First Edition (December 2000)

This edition applies to LDAP Version 3 compliant directories.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

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

Contents

Preface

This redbook is intended for systems and network administrators, support personnel and consultants who are faced with today‘s directory management challenges.

This book explains directory and LDAP concepts, and deals with some basic and advanced LDAP tasks, such as referrals and schema extensibility. We demonstrate ways to tackle these issues by taking a scenario-based approach, using some leading directory products available: the IBM SecureWay Directory, Lotus Domino, and Microsoft’s Active Directory.

This redbook will help you if:

- You have an LDAP Directory and you are looking to implement an Enterprise Directory.
- You have Active Directory and you looking to implement, integrate or move to the IBM SecureWay Directory.
- You have Lotus Domino and you want to leverage Domino’s directory as a base for populating other directories.
- You have an Enterprise Directory and you are looking to add Active Directory to your environment.
- You want to exchange, replicate or synchronize data between the IBM SecureWay Directory and Active Directory.

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, Raleigh Center.

Harri Levanen is an Advisory Campus LAN specialist at the International Technical Support Organization, Raleigh Center. He has six years of experience in the networking area. His areas of expertise include ATM switches, LAN switches and network management. Before joining the ITSO in early 1999, Harri worked in IBM Global Services in Finland.

Bernard Freund is a senior consultant and partner with Enterprise Consulting Partners in Brazil. He has 10 years of experience in Information Technology consulting, and six years of experience in X.500 and LDAP directories. He holds a degree in Computer Sciences from Universidade Federal do Rio Grande do Sul. His areas of expertise include Microsoft infrastructure products, directory management, security and messaging. He holds 35 Microsoft certifications, including Active Directory Design and Active Directory Implementation and Support. He is a Microsoft Certified Systems Engineer+Internet, Microsoft Certified Database Administrator and Microsoft Certified Trainer. He also holds several directory management-specific certifications, including Certified Directory Management Specialist and Certified Directory Management Developer, from Fastlane Technologies. Enterprise Consulting Partners is a services and product sales organization that focuses on Information Technology infrastructure.

Hani Mansi is an IT Security Specialist with the Security & Privacy Services group in IBM Global Services in Canada. He has nine years of experience in the Information Technology field and is a graduate of the University of Alberta. Hani has worked as a Systems Programmer on OS/390 and VM operating systems and as a Senior Technical Analyst for client server environments. Hani's areas of expertise include implementing OS/390 images, overseeing the design, security and implementation of Microsoft domains and Enterprise Resource Planning (ERP) solutions as well as troubleshooting performance issues. He holds Microsoft certification for Windows NT 4.0 Server and Workstation and is trained on Cisco routers and switches.

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

Heinz Johner, IBM Switzerland, for his LDAP insight and by providing the redbooks *Understanding LDAP*, SG24-4986 and *LDAP Implementation Cookbook*, SG24-5110.

Pat Fleming, IBM USA, Rochester, for his contributions on schema extensibility and compatibility

Mike Ebbers, International Technical Support Organization, Poughkeepsie Center, for his contributions on Domino by providing the redbook *Getting the Most From Your Domino Directory*, SG24-5986.

Tatsuhiko Kakimoto, International Technical Support Organization, Raleigh Center, for his extensive help on OS/390

George Baker, International Technical Support Organization, Raleigh Center, for his help with the IBM ePerson schema and schema compatibility

Paul de Graaff, International Technical Support Organization, Poughkeepsie Center, for providing the redbook *OS390 Security Server 1999 Updates Technical Presentation Guide*, SG24-5627

Abbas Farazdel, IBM Poughkeepsie, for providing the redbook *Exploiting RS 6000 SP Security Keeping It Safe*, SG24-5521

Linda Robinson, Margaret Ticknor, Gail Christensen
International Technical Support Organization, Raleigh Center

Bob Haimowitz
International Technical Support Organization, Poughkeepsie Center

Mike Haley
Formerly International Technical Support Organization, Raleigh Center

Diane Pearce
IBM Canada

Mark McConaughy
IBM Austin

Comments welcome

Your comments are important to us!

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

- Fax the evaluation form found in “IBM Redbooks review” on page 227 to the fax number shown on the form.
- Use the online evaluation form found at `ibm.com/redbooks`
- Send your comments in an Internet note to `redbook@us.ibm.com`

Chapter 1. Introduction of the components

People and businesses are increasingly relying on networked computer systems to support distributed applications. These distributed applications might interact with computers on the same local area network (LAN), within a corporate intranet, or anywhere in the world on the Internet. To improve functionality, provide ease of use, and 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 to prevent unauthorized modification or the disclosure of private information. Security, however, is not the only consideration when applying a service policy to a piece of communication. The quality of service to be delivered is another major element, and directories today are capable of holding millions of objects.

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 is gaining 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 increasingly incorporated into applications. For example, the two most popular Web browsers, Netscape Navigator/Communicator and Microsoft Internet Explorer, support LDAP as a base feature.

This chapter defines standards and terminologies involved in accessing directories. This provides a framework for discussing common directory definitions, the particularities of implementation and the part that a corporate directory can play in an integrated environment.

1.1 What is a directory?

A directory is a listing of information about objects arranged in some order and 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 attribute 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 particular directory applications. 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. On the other hand, if the name of a particular individual attribute 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 a range of criteria, not just by a single predefined set of categories.

1.1.1 Why is a directory service important?

Without realizing it, we all use directory services, whether on the Internet or at work when we need to get some information. When you type in a URL on a Web browser, such as `http://www.s390.ibm.com`, this must be translated into an IP address via the Domain Name System (DNS). Thus, DNS is a directory that you use.

The use of directory services is everywhere. RACF, which is one of the SecureWay Security Server components in the OS/390 environment, is nothing more than a security directory (database) that you use to identify yourself and check your access to data.

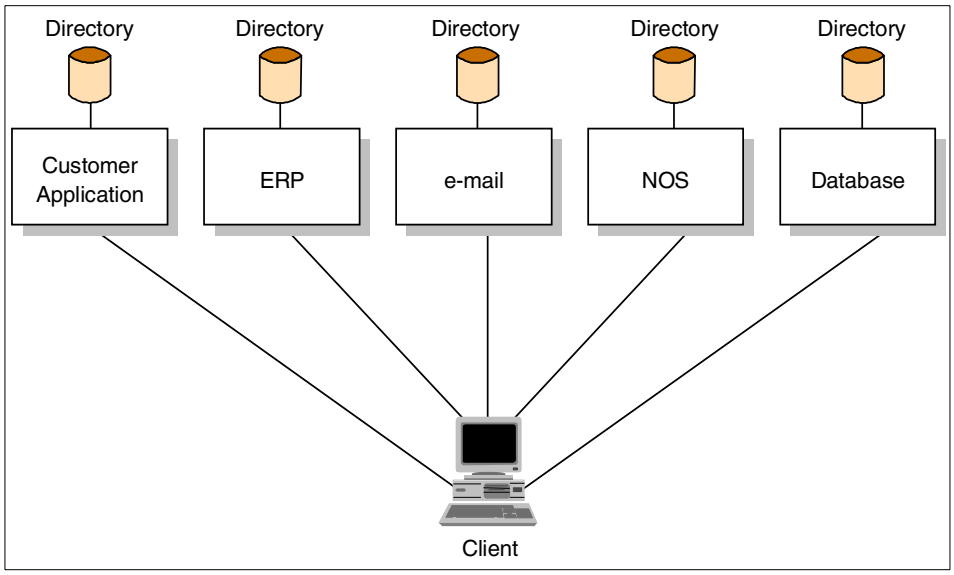

Figure 1. The directory problem

Several other examples exist. One that has generated a lot of interest is Directory-Enabled Network (DEN). This is an example of LDAP being viewed as a factor in easing the management of many different components of distributed systems. It may also provide the capability to centralize the management of these distributed systems without reducing security or increasing complexity. For more on this, see Chapter 7, “Future” on page 143.

Today, the developers of directory-enabled applications are faced with a problem. What if they cannot assume that a directory service will exist in all environments? If there is a directory service, it might be specific to a certain network operating system (NOS) or e-mail system, making the application non-portable. Can the existing directory service be extended to store the type of information needed by the application? Because of these concerns, application developers often take the approach of developing their own application-specific directory.

To avoid this obvious waste of time and energy (by reinventing the wheel on every application), directory standards were created.

1.1.2 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. The results of the read or write are then returned to the requesting application (see Figure 2).

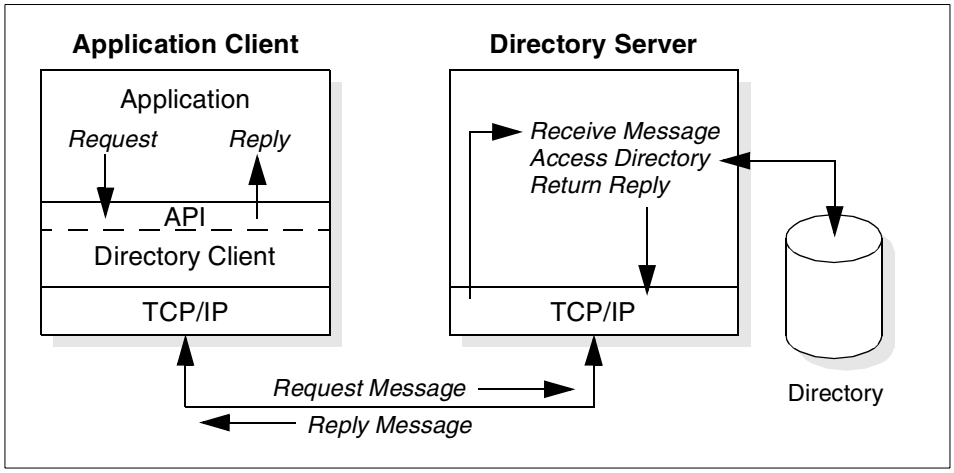

Figure 2. Directory client/server interaction

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.

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 might or might 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 and JDAP APIs for C language, and ways to access the directory from a Java application using JNDI. The client is not dependent upon a particular implementation of the server, and the server can implement the directory however it chooses.

1.1.3 Distributed directories

The terms *local*, *global*, *centralized*, and *distributed* are often used to describe a directory or directory service. These terms mean different things to different people in different contexts. In this section, these terms are explained as they apply to directories in different contexts.

In general, *local* means something is close by, 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. Like 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, public encryption keys, 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 allow the users to refer LDAP requests to either the same or different name

spaces stored in a different (or the same) 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 can often be controlled to affect the performance and availability of the directory. For example, a distributed and replicated directory might perform better because a read request can be serviced by a nearby server. A centralized directory may be less available because it is a single point of failure. However, a distributed directory might be more difficult to maintain because multiple sites, possibly under the control of multiple administrators, must be kept up-to-date and in running order.

The design and maintenance of a directory service can be complex, and many trade-offs may be involved. You can see more detailed information about the topic in the following chapters.

1.1.4 Directory security

The security of information stored in a directory is a major consideration. Some directories are meant to be accessed publicly on the Internet, but any user should not necessarily be able to perform any operation. A company's directory servicing its intranet can be stored behind a *firewall* to keep the general public from accessing it, but more security control may be needed within the intranet itself.

For example, any intranet user should be able to look up an employee's e-mail address, but only the employee themselves or a system administrator should be able to change it. Members of the personnel department might have permission to look up an employee's home telephone number, but their co-workers might not. Depending on the confidentiality of the data, information may need to be encrypted before being transmitted over the network. A security policy defines who has what type of access to what information, and is defined by the organization that maintains the directory.

A directory should support the basic capabilities needed to implement a security policy. The directory in this case is one of the components by which a security mechanism is put in place for the whole network. It is also one of the network resources that itself needs protecting.

First, a method is needed to authenticate users. Authentication verifies that users are who they say they are. A user name and password is a basic authentication scheme. Once users are authenticated, it must be determined if they have the authorization or permission to perform the requested operation on the specific object.

Authorization is often based on access control lists (ACLs). An ACL is a list of authorizations that may be attached to objects and attributes in the directory. An ACL lists what type of access each user or a group of users is allowed or denied. In order to make ACLs shorter and more manageable, users with the same access rights are often put into security groups. Table 1 shows an example ACL for an employee's directory entry.

Table 1. Example ACL for an employee's directory entry

User or Group	Access Rights
owner	read, modify (but not delete)
administrators	all
personnel	read all fields
all others	read restricted

1.1.5 Users, platforms, and networks

An understanding of users (what a user is, where the information about a user is stored, how it is used, and so on) is key to understanding the general relationship between white page directory services and platform operating systems.

Starting with a white page directory service, there are two obvious ways to think about users:

- A user may be an LDAP directory user. LDAP sessions begin with a bind operation. For cases other than unauthenticated access, a bind request must supply a distinguished name (DN) and a password or another type of credential. So, a user ID must be defined in the directory server (the user must have a DN) in order to be authenticated to access the directory.
- A user may have a white pages entry in an LDAP directory. The common white pages or phone book application will contain information about

people, in most cases many people. There can be, but does not have to be, a relationship between a white pages entry and an LDAP directory user entry; that is, everyone listed in the white pages directory may not have access to the directory. The inverse may also be true; people not listed in the directory may access the data as, for example, when internal users access a directory that stores customer *person objects*.

Most operating system platforms provide local or remote access for users; so, a user ID can also be handled in the following ways:

A user may have a system-defined user ID. The user ID must be defined on the system platform in order to be authenticated so they can be a user of that platform for access to system services, such as file and print sharing. Such user IDs are usually accompanied by a password, or, with increasing importance, by a digital certificate signed by a known Certificate Authority (CA).

Historically, multi-user interactive applications and subsystems used the platform-level user definitions on the hosted platform to authenticate user access to transactions, programs and data. While single sign-on and the overall security architecture are beyond the scope of this guide, it is sufficient to say that, as the number of systems in a network continues to grow and applications themselves are distributed across multiple systems, the strict application-to-host system user affinity is insufficient. So, many applications have developed their own private user and authentication files.

A user may be an application user. This user ID must be defined/enrolled in the specific application in order to be authenticated and have access to that application and its resources.

It is important to understand this current topology, with its mix of semantics, syntax, and backing stores for user information, in order to understand the context of users within a distributed environment.

1.1.6 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, for example, general-purpose relational databases. One special characteristic of directories is that in general 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.

Directories must be able to support high volumes of read requests, so 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 database, on the other hand, needs to support applications such as airline reservations and banking with high update volumes.

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 could contain the network address of a print server. The print server could be queried to learn 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 important difference between directories and general-purpose databases is that directories may not support transactions. 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 databases usually support such transactions, which complicates their implementation.

Directories deal mostly with read requests, so the complexities of transactions can be avoided. If two people exchange offices, both of their directory entries need to be updated with new phone numbers, office locations, and so on. If one directory entry is updated, and then other directory entry is updated there is a brief period during which the directory will show that both people have the same phone number. Because updates are relatively rare, such anomalies are considered acceptable.

In contrast to directories, general-purpose databases must support arbitrary applications such as banking and inventory control, so they allow arbitrary collections of data to be stored. On the other hand 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 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. LDAP directories, on the other hand, use a simplified and optimized access protocol that can be used in slim and relatively simple applications.

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

1.2 Directory 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 of the independent major standardizations efforts are LDAP and X.500.

1.2.1 X.500 - the directory service standard

During the 1980s, the growth in implementations of wide area network communication forced the deployment of a new set of networking protocols called open system interconnection (OSI). OSI presented a seven-layer model of communications. Part of the standards developed by CCITT is a definition of generic directory service. CCITT defined the first X.500 standard in 1988, which then became ISO 9594, Data Communications Network Directory, Recommendations X.500/X.521 in 1990, though it is still commonly referred to as X.500.

Although the context and the content of both are the same, the two organizations use different terms: the ISO describes ISO/IEC 9594 as a multi-part standard, whereas the ITU-T refers to X.500 as a series of recommendations. To avoid confusion, we adopt the term *specification* which is accepted by both ISO and ITU-T.

The directory specifications are available in four editions:

1988 edition: This is the first edition and is issued as the multi-part standard ISO/IEC 9594:1990 and the CCITT X.500 (1988) Series of Recommendations. This edition specifies services, protocols and procedures necessary for basic directory operations, information models for how information is structured, and it specifies some commonly usable information objects. In addition, it provides a common framework for general authentication techniques.

1993 edition: This second edition is issued as ISO/IEC 9594: 1995 and as ITU-T X.500 (1993). Added functions include shadowing of directory information, replication, access control and the expansion of the information model. Administrative capabilities are built in, too.

1997 edition: This edition adds a feature called contexts, which allows information to be distinguished according to the context in which it is being accessed. Another important addition is the provision of OSI management of the directory. It also has added and extended important security features.

2001 edition: This edition include higher Internet (TCP/IP) integration.

Further extensions of the directory specifications are currently in progress, and will include:

- The new service concept, which allows an administrator authority to tailor the service provided to the user through elaborate service administration tools.
- The hierarchical group feature, which allows establishment of a hierarchical structure independent of the DIT structure.
- The family of entities feature, which allows information about an object to be structured in a more logical way and allows powerful matching and rules for return of information.
- The mapping-based searches features, which allows search requests to provide a mapping between the world as the user sees it and the world as it is reflected by the directory.

For more information about the defined standards see Appendix D, “Standards” on page 175.

X.500 organizes directory entries in a hierarchical name space capable of supporting large amounts of information. For example, in the general case, a name consists of several components reflecting hierarchy. This naming tree

is called the directory information tree (DIT), as a directory entry is associated with each vertex of this tree, where the entry holds information about the object having the corresponding name (see 1.2.1.6, “The X.500 directory structure” on page 14).

The following sections explain the different models and protocols regarding X.500.

1.2.1.1 The X.500 model

The X.500 directory system consists of three principal components:

- The Directory Information Base (DIB)
- Directory User Agents (DUAs)
- Directory Service Agents (DSAs)

If a user will get information stored in a directory they get access over a DUA, which represents the “client” side of the directory services. Users can be either people or application programs.

1.2.1.2 The X.500 protocols

To define the connection between the different directory functional components, two protocols were developed (see Figure 3).

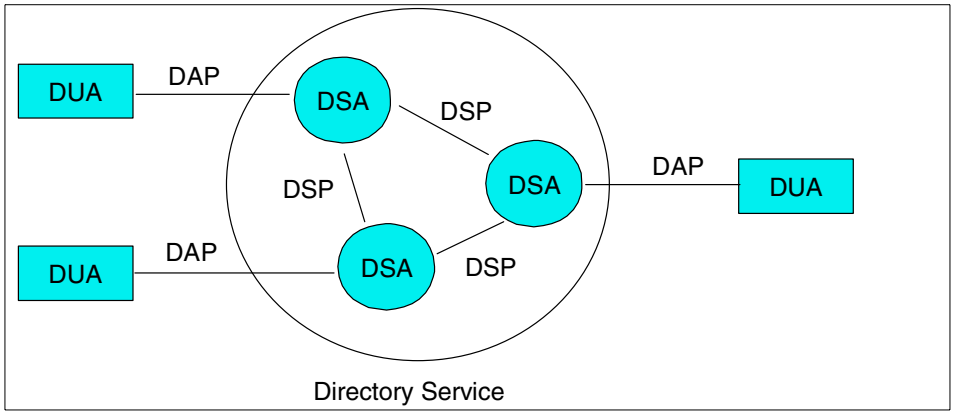

Figure 3. Distributed directory model

The Directory Access Protocol (DAP) defines how DUAs get access to the information stored in DSAs.

If the information stored in the directory grows it might be necessary to split the DIB in multiple DSAs. A Directory Service Protocol (DSP) is used between two DSAs to query user information lookups over multiple DSAs.

A set of one or more DSAs and zero or more DUAs managed by a single organization may form a Directory Management Domain (DMD). A DMD may be an Administration DMD (ADDMD) or a Private DMD (PRDMD), depending on whether or not it is being operated by a public telecommunication organization or by service provider.

There are more protocols in X.500, but those are the basic ones.

1.2.1.3 Overview of the Directory Access Protocol (DAP)

Here's how the DSAs cooperate to provide a user service.

Bind and unbind operations

The directory bind operation is the operation that establishes a connection between the DUA and the DSA. Conversely, the directory unbind operation is the operation that closes the connection between the two.

Interrogation operations

There are four interrogation operations:

- The Read operation reads the information from one specific directory.
- The Compare operation compares the user-presented attribute value with those actually existing in the entry.
- The List operation lists the immediate subordinates of an entry.
- The Search operation is used to search portions of the DIT and to return selected information about selected entries.

In addition to these, the Abandon operation allows the user to abandon any of the above interrogation operations, if they are no longer interested in obtaining the result.

Modification operations

Several different operations to modify entries in a directory were defined in the 1988 standards; most of them were removed in the 1993 edition of the standards:

- The AddEntry operation
- The RemoveEntry operation
- The ModifyRDN operation
- The ModifyDN operation

- The ModifyEntry operation

1.2.1.4 Overview of the Directory System Protocol (DSP)

This protocol is very similar to DAP. It is used between different DSAs to answer a user's DAP queries.

The first step must be the bind operation, which works in the same way as the DAP bind operation.

The operations that flow between the DSAs are called chained operations. Each of the DAP operations has a corresponding DSP chained operation, and each result has a corresponding chained result.

1.2.1.5 Overview of the Directory Information Shadowing Protocol

The Directory Information Shadowing Protocol (DISP) is used to transfer information from the shadow supplier to the shadow consumer. The DISP consists of three operations plus the DISP shadow bind and unbind operations, which are used to open and close connections. Either of two operations may be used to prepare both of the DSAs to send and receive the shadowed information. The third operation is used to actually transfer the shadowed data. Either the Coordinate Shadow Update operation, which is sent from the shadow supplier to the shadow consumer, or, alternatively, the Request Shadow Update operation, which is sent from the shadow consumer to the shadow supplier, may be used to synchronize the DSAs for the next batch of updates.The Update Shadow Operation is used to actually transfer the batch of updates from the supplier to the consumer. Each batch of updates is timestamped by the supplier, so that they can be uniquely identified. This aids discovery of the cause, in the event of a failure.

Note

One of the most broadly used applications of the X.500 standard are X.509 certificates. To learn more on this subject, see Appendix I, "X.509 certificates" on page 205.

1.2.1.6 The X.500 directory structure

The information stored in the DIB is structured as a hierarchical information tree, called Directory Information Tree (DIT). Each level of the DIT is called an *object class*; all objects in the same class share the same attributes (see Figure 4 on page 15).

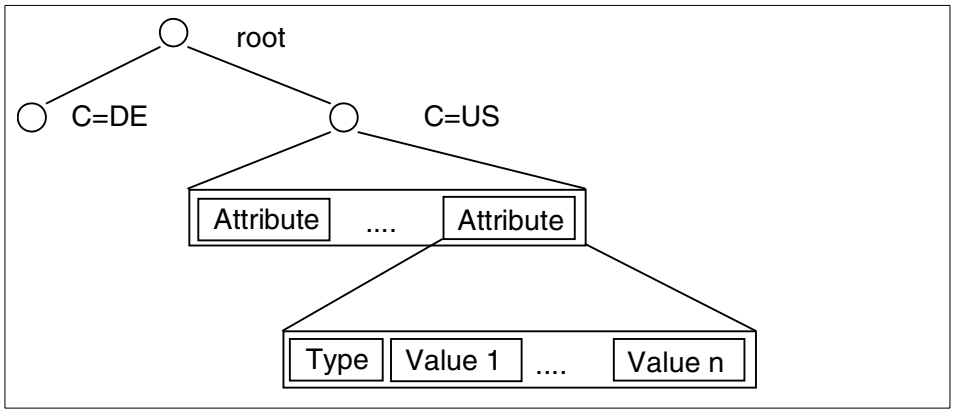

Figure 4. Directory entry structure

A DIT address is made up of a sequence of Relative Distinguished Names (RDNs), which are ordered sequences of attributes.

The example shown in Figure 5 explains the structure of DITs and the special terms.

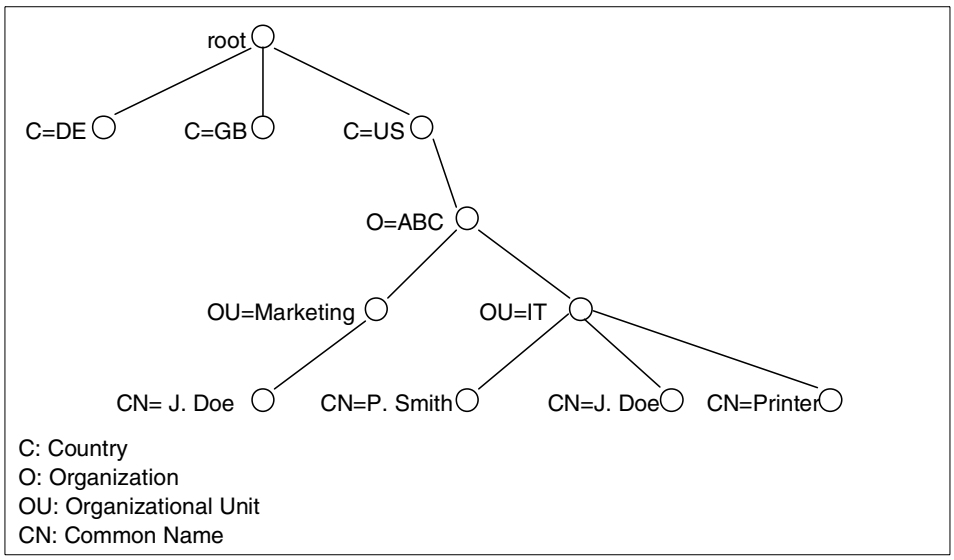

Figure 5. Example of a directory information tree

The example shows a DIT with entries for three countries. The name space for US is divided into an organizational level name space with different user levels.

If you look at the hierarchy for the entry of P. Smith you can see the following naming attributes, which represents the following Distinguished Name:

- Country: US
- Organization: ABC
- Organizational Unit: IT
- Common Name: P.Smith

There are two entries with the common name J.Doe, but they can be distinguished by their full RDNs:

- {C=US, O=ABC, OU=Marketing, CN=J. Doe}
- {C=US, O=ABC, OU=IT, CN=J. Doe}

This shows you that the directory requires the full Distinguished Name to locate an entry.

Users typically do not need to type in the full name. Software solutions for DUA can expand it automatically to the full name and provide selections among similar entries.

Different vendors use different structures of DITs. For example, Microsoft Active Directory uses domain component (dc) instead of country (c) and organization (o). Some deployments of DITs prefer to use DNS-based naming, which is available with most directory products.

1.2.1.7 Operation of the X.500 model

The DUA interacts with the directory by communicating with one or more DSAs. There are different ways of request-handling, as illustrated in Figure 6 through Figure 10.

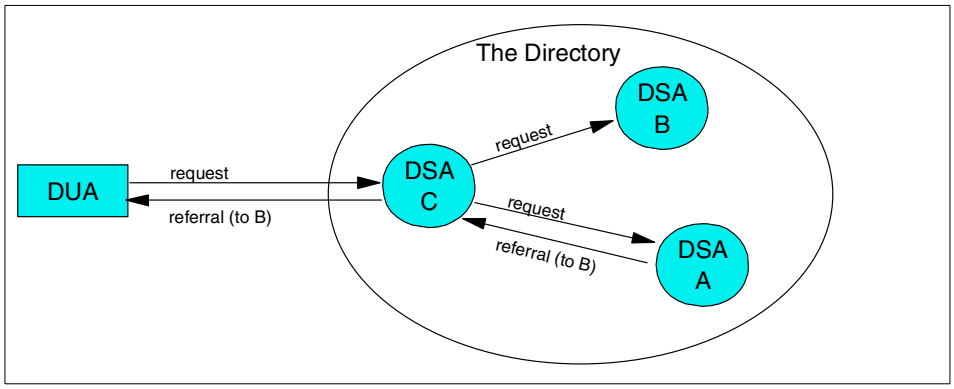

Figure 6. X.500 referrals (1)

The referrals from DSA A will be received by DSA C, which is responsible for conveying every request to the DSA B or conveying the referral to the originating DUA.

If DSA C returns the referral to the DUA, the request to DSA B will not occur.

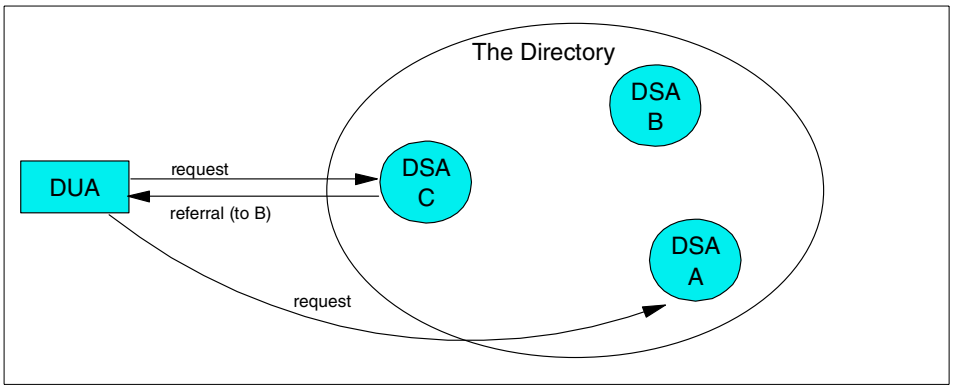

Figure 7. X.500 referrals (2)

Referrals provide a way for servers to refer clients to additional directory servers. With referrals you can:

- Distribute namespace information among multiple servers
- Provide knowledge of where data resides within a set of interrelated servers
- Route client requests to the appropriate server

Using referrals has several advantages. Referrals let you:

- Distribute processing overhead, providing primitive load balancing
- Distribute administration of data along organizational boundaries
- Provide potential for widespread interconnection, beyond an organization's own boundaries

In the case shown in Figure 7, the DUA receives the referral from DSA C and is responsible for reissuing the request directly to DSA A (which is named in the referral from DSA C).

If a DUA is connected directly to one DSA and this DSA cannot respond, it will send a request to another DSA. This process is called *chaining*.

It is possible to pass the request through several DSAs before a response is returned. This is shown in Figure 8.

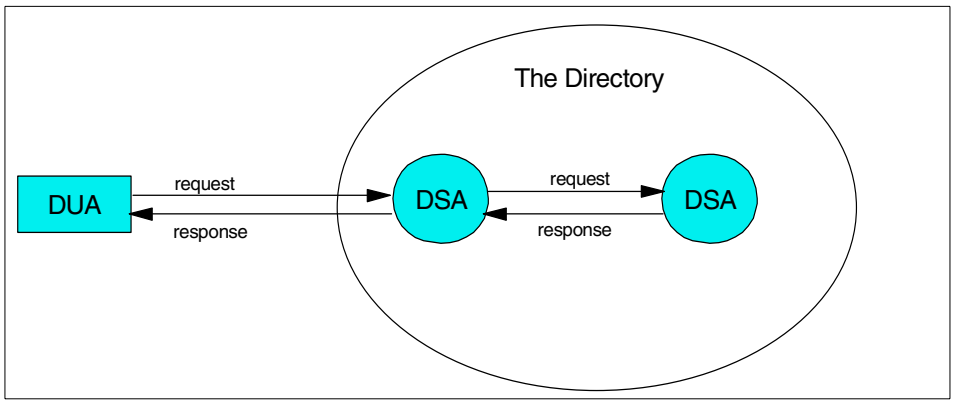

Figure 8. Uni-chaining

Forwarding the request to two or more DSAs is called multi-chaining; this is illustrated in Figure 9 on page 19.

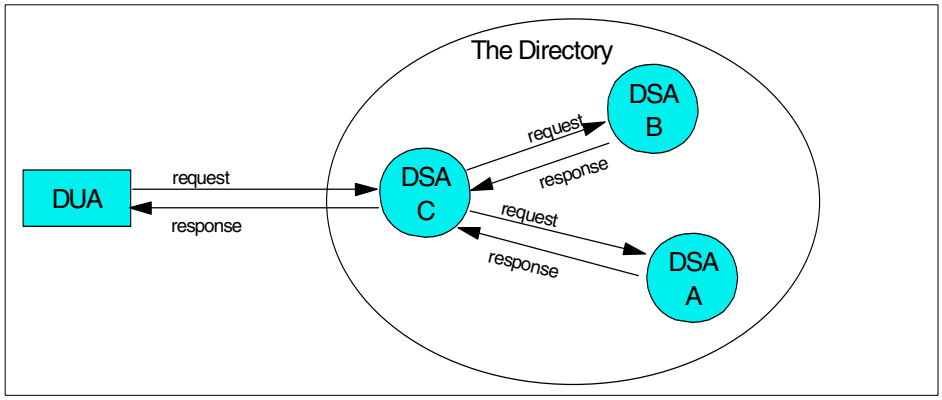

Figure 9. Multi-chaining

In a directory environment you can find all the approaches. A hybrid approach that combines functional interactions may be needed to satisfy a request. An example is shown in Figure 10.

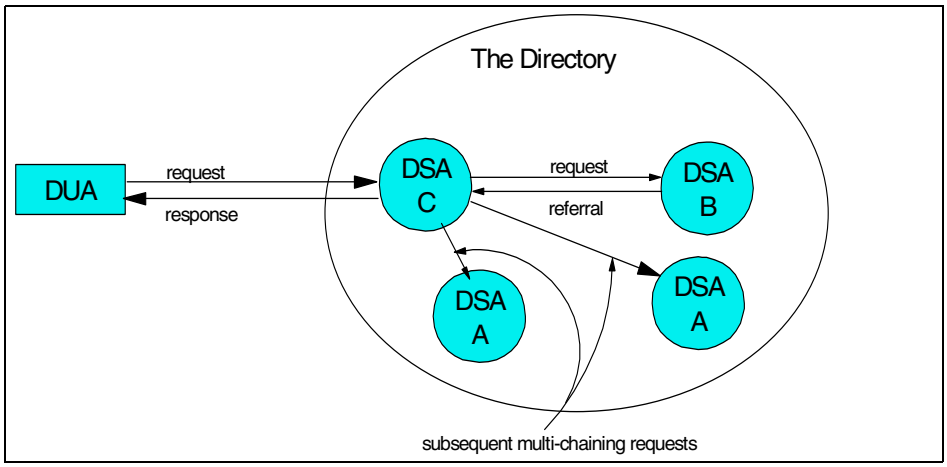

Figure 10. Mixed modes hybrid approach

1.2.2 LDAP

X.500 is based on the client/server model of distributed computing and LDAP represents the client implementation of that model. LDAP evolved as a lightweight protocol for accessing information in X.500 directory services. It has since become independent of X.500; servers that specifically support the LDAP protocol rather than the X.500 Directory Access Protocol (DAP) are now common. 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:

- LDAP runs over TCP/IP rather than the OSI protocol stack. TCP/IP is less resource-intensive and is much more widely available, especially on desktop systems.
- The functional model of LDAP is simpler. It omits duplicate, rarely used and esoteric features. This makes LDAP easier to understand and to implement.
- LDAP uses strings to represent data rather than complicated structured syntaxes such as ASN.1 (Abstract Syntax Notation One).

Several standards in the form of IETF RFCs and drafts exist for LDAP. More information and a brief description of these standards (RFCs) can be found in Appendix D, “Standards” on page 175.

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

1.2.2.1 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 (search, modify, delete, and so on), 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-based 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:

- 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. 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.

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

Although it is not defined by the LDAP protocol itself, there is a well-known LDAP API (application program interface) that allows applications to easily interact with LDAP servers. The API can be considered an extension to the LDAP architecture. The C language LDAP API associated with LDAP V2 is an informational RFC. The update to it has not yet progressed to RFC status. However it has achieved de facto standard status because it is supported by all major LDAP vendors. 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 an X.500 model (see 1.2.1, “X.500 - the directory service standard” on page 10). The directory stores and organizes data structures known as *entries*.

A directory entry describes some object. An object class is a general description 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:

- 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

Objects can be derived from other objects. This is known as *subclassing*. For example, suppose an object called *person* was defined that included a surname and so on. An object class *organizationalPerson* could be defined as a subclass of the person object class. The organizationalPerson 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 organizationalPerson object class. One special object class, called top, has no superiors. The top object class includes the mandatory *objectClass* attribute. Attributes in the top object class 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).

1.2.2.2 LDAP models

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.

For more information about LDAP models see *Understanding LDAP*, SG24-4986, and *LDAP Implementation Cookbook*, SG24-5110.

1.2.2.3 LDAP: protocol or directory

LDAP defines a communication protocol. That is, it defines the format of messages used by a client to access data in a directory service that listens for and responds to LDAP requests. LDAP does not define the directory service itself, yet people often talk about LDAP directories. Others say LDAP is only a protocol, that there is no such thing as an LDAP directory. What is an LDAP directory?

An application client program initiates an LDAP message by calling an LDAP API. But an X.500 directory server does not understand LDAP messages. The LDAP client actually communicates with a gateway process (also called a proxy or front end) that forwards requests to the X.500 directory server (see Figure 11). This gateway is known as an LDAP server. It services requests from the LDAP client. It does this by becoming a client of the X.500 server.

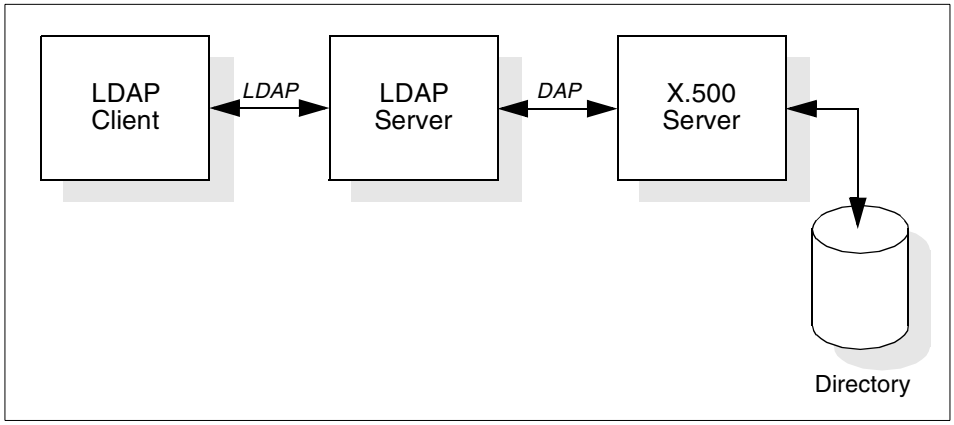

Figure 11. LDAP server acting as a gateway to an X.500 server

As the use of LDAP grew and its benefits became apparent, people who did not have X.500 servers or the environments to support them wanted to build directories that could be accessed by LDAP clients. So, why not have the LDAP server store and access the directory itself instead of only acting as a gateway to X.500 servers, as shown in Figure 12? This eliminates any need for the OSI protocol stack. Of course this makes the LDAP server much more complicated since it must store and retrieve directory entries. These LDAP servers are often called stand-alone LDAP servers because they do not depend on an X.500 directory server. Since LDAP does not support all X.500

capabilities, a stand-alone LDAP server only needs to support those capabilities required by LDAP.

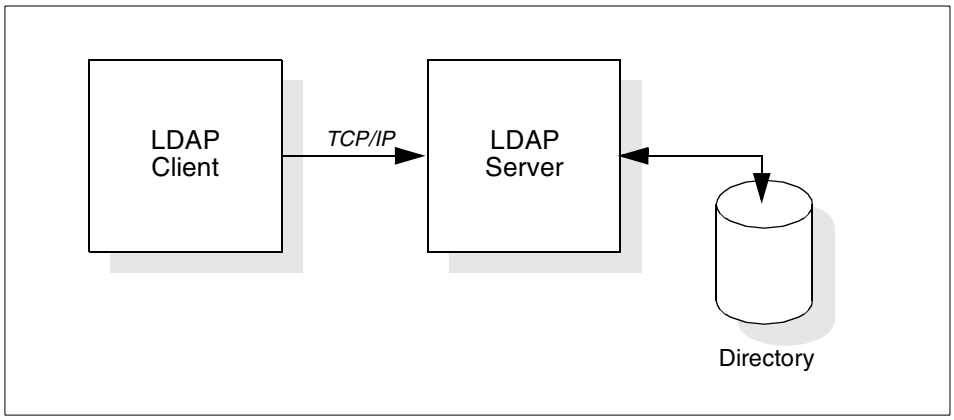

Figure 12. Stand-alone LDAP server

RFC 1777 (LDAP Version 2) discusses providing access to the X.500 directory. RFC 2251 (LDAP Version 3) discusses providing access to directories supporting the X.500 model. This change in language reflects the idea that an LDAP server can implement the directory itself or can be a gateway to an X.500 directory.

From the client's point of view, any server that implements the LDAP protocol is an LDAP directory server, whether the LDAP server is one that supports the X.500 model or is a gateway to an X.500 server.

The directory that is accessed can be called an LDAP-based directory, whether the directory is implemented by a stand-alone LDAP server or by an X.500 server.

1.3 Enterprise directory

The *enterprise directory* within an organization is classically envisioned to be the authoritative repository of heterogeneous corporate information that can be used by a variety of applications. While this definition has been relatively consistent over time, the technology associated with it has changed.

The initial promise of enterprise directory functionality was based on an anticipated widespread adoption of the X.500 directory standard. More recently, with the emergence of LDAP as the de facto commercial standard for directory services, the IETF and related committees have focused on

ensuring that this protocol has the characteristics needed to be effectively used in this central infrastructure role.

The features required of an enterprise directory include standards implementation, extensibility, manageability, security and scalability.

The current standard of choice is LDAP. Its latest iteration is LDAP V3. Major industry participants, including Lotus, IBM, Microsoft, Oracle, Novell, Sun/Netscape, Nexor, Siemens, Computer Associates and others have embraced it and have placed LDAP-based directory offerings on the market. Also important is the industry's efforts to ensure that these standards-based implementations are interoperable. For example, an industry forum called The Directory Interoperability Forum, `http://www.opengroup.org/directory/`, invites technical representatives from various directory vendors to periodic meetings. During these meetings interoperability issues are identified and corrected. These corrections are then incorporated into the directory products.

Extensibility is the ability of a directory product to adapt to the requirements of the organization it is supporting. This is generally discussed in terms of extensions to the LDAP schema. This is a set of rules which define the nature of the information items and how they interact. In an enterprise environment, this set of directory-defining rules must be not only extensible, but also capable of being accessed by applications seeking to use the information stored in the directory. The LDAP standards provide a mechanism for defining and publishing the initial schema and for extending that schema if necessary. So an enterprise directory should not only comply with the LDAP standards for populating, locating, and returning information in the directory, but also define and extend the directory schema.

Manageability refers to the capability of an organization to maintain control over the contents of the directory, its directory's performance and its ability to monitor, troubleshoot, and correct the various directory processes in use. Here again, the implementation of a standards-based directory can provide concurrent flexibility in the tools used to manage that directory and the enterprise directory environment.

Security is generally viewed in the context of maintaining control over access to the information stored in the directory. Since an enterprise directory is the authoritative repository of enterprise information, it is critical to maintain the ability to tightly control which specific individuals and groups are permitted to create, modify, and delete information in it. Furthermore, it is important to provide a granular level of control. This can range from a simple "has access or does not have access" (if an individual has access to the directory they are

allowed to change anything they want), to complex controls imposed at the individual attribute level. Currently there is no component of the LDAP standard that addresses access control, so each vendor has implemented its own mechanisms. However, such a standard is being developed and should begin to appear in products during 2001.

Scalability of a directory is the least standard-dependent and most vendor-specific area of enterprise directory technology. Vendors have taken different approaches to achieving enterprise scalability for their products. Enterprises need to exercise caution, as they do in any performance measurement, to ensure that they understand two critical things when judging enterprise directory scalability. The first of these is to understand how the vendor's scalability claims are developed and supported, in order to understand how well that environment matches their own. The second is to have a realistic understanding of their own scalability needs. This is not an area where getting the biggest is necessarily the best. A good match between requirements and product capabilities, with a sufficient margin for realistic growth, is the most effective solution.

A final aspect of an organization's enterprise directory, just as critical as the others, is interoperability. While the definition of an enterprise directory implies a single directory construct, the reality for the vast majority of enterprises is that they will continue to function in a multi-directory environment for the near- to mid-term. So the ability to interoperate with the other directory stores in the organization is a key characteristic of an enterprise directory. Classically this requirement is met either through a directory synchronization mechanism or through metadirectory technology.

1.3.1 Directory synchronization

Directory synchronization traditionally has been oriented toward messaging infrastructures, because this was where the early major requirements were formulated. We use this as an example to show how directory synchronization works.

Most e-mail systems today provide a directory that enables users to look up and select mail recipients. However, because mail systems have developed independently, a user of one type of mail system cannot look at entries in the directory of a different mail system.

This limitation, and the need for enterprises to form a unified messaging system, indicates that a mechanism for synchronizing name and address information across heterogeneous LAN-based or mainframe-based mail systems is required.

The same structure can be used for enterprise directories, shown in Figure 13. In this example we have three different local directories: a departmental mail system, a LAN-based system and a mainframe system. The directory synchronization handles three steps:

1. It sends a set of information from the local directory to the enterprise directory (uploading from client to server).
2. Backbone (or background) replication inside the enterprise directory service occurs.
3. The changes are sent from an enterprise directory to the local directories (downloading from server to client).

Because of this scheduled process it is possible to have all the necessary information about directory entries in all systems in the heterogeneous environment.

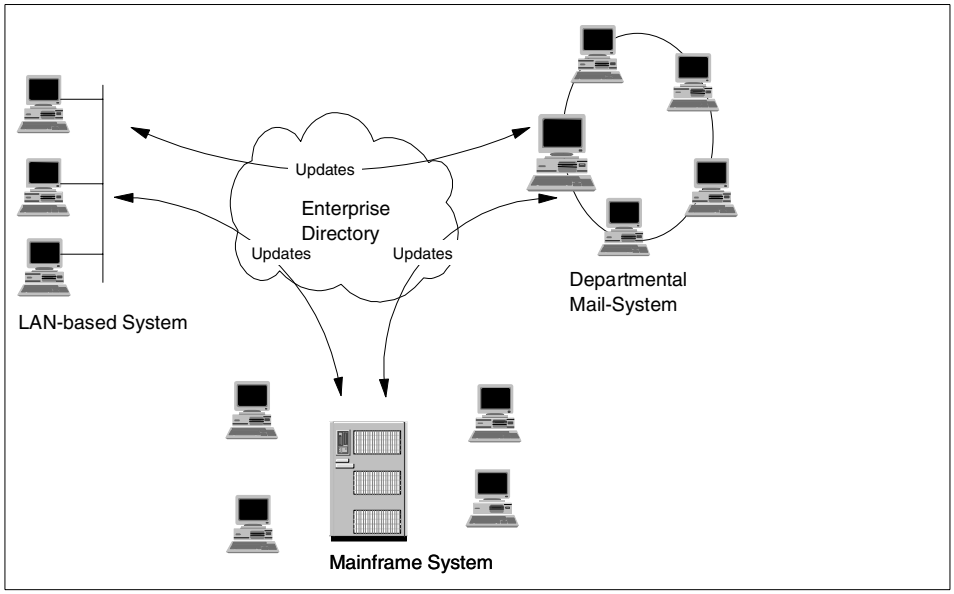

Figure 13. Directory synchronization

The process of Directory synchronization replicates directory information across a heterogeneous network. It make it possible to have information in each directory that is used by the users in the enterprise network.

There are two common methods available to synchronize existing directories:

- Synchronization based on having the same user name and password in all directories (point-to-point synchronization or n-way synchronization)
- Integration of existing directories into one metadirectory (one-way and two-way synchronization)

The first method, based on the use of the same user name and password, requires that a synchronization mechanism exist between the different directory resources and types.

To do this, a common identifier (unique key) is required on all systems. This unique key can be as simple as a personal number, a social security number, or any other custom-made string, as long as it is unique and is supported by all systems. Based on the systems used, there will be cases where multiple instances of user credentials, as well as aliases, are required.

Common attributes or attribute mapping tables will ensure that during the synchronization process between the systems all unique user information is synchronized. In a few cases, directory mapping tables will be used to make it possible for the end users to match certain attributes to custom-defined fields.

An additional method uses a central directory to store all relevant information such as user names, passwords, and group names, as well as user certificates. This can be any directory store capable of storing all the required information and with access to the connecting systems and applications.This is also a common strategy for metadirectories (see 1.4, “Metadirectory” on page 29).

The challenge is to integrate all the existing systems and applications to authenticate through this central directory. Based on the available tools and APIs, this can result in multiple prompts to the end user for authentication. The typical tools used for this will focus on application integration. Some of the tools and methods used for this purpose are C/C++, Visual Basic, LDAP APIs, Java, ADSI (Microsoft Active Directory Services Interface), JNDI (Java Naming and Directory Interface), and LEI 3.0 (Lotus Enterprise Integrator).

1.3.2 Organizational units (OUs)

The organizational units are the most widely used container objects, and they are used to organize other objects, such as other OUs and leaf objects.

Several different models can be used for creating an OU structure: geographic, organizational hierarchy, business process, network administration.

This choice is an extremely important one, and it should not be taken lightly, because this will affect dramatically your Total Cost of Ownership (TCO) and the general enterprise policies. The OU structure can be very difficult to change after it is implemented, so we suggest that you plan carefully and that you do extensive pilots *before* implementing your OU structure.

1.4 Metadirectory

There are several interpretations of the term metadirectory, but it is usually used to describe a central enterprise directory that is able to synchronize with multiple directory systems and manage the relationship between heterogeneous namespaces. Metadirectory is a tool for accessing distributed data from disparate sources. The term and the concept were originally introduced in a report entitled "Directory Services Strategic Overview: Meta-Directory Services" produced by the Burton Group in February 1996, and revised in 1998.

Despite the growing recognition of the common requirement for metadirectory solutions, there are as yet no standards for metadirectory, or any unequivocal consensus among vendors as to what exactly constitutes a metadirectory. Until a definition is achieved, metadirectory will remain a conceptual term based upon a broad set of user requirements for an integrated, distributed directory service, and used to describe a range of products, services and environments. In general, metadirectory products are batch or event-driven middleware providing management capabilities across disparate directory and database systems; and a metadirectory deployment usually provides a coordinated administration system and a single logical namespace across a heterogeneous networking environment.

Two opposite directions

Some users of metadirectories wish that there was a clear separation between the directory and the metadirectory with a possibility to choose a different directory or metadirectory. The vendors tend to tie their directory and metadirectory product together and some even say that the product separation between directory and metadirectory will disappear, since vendors will integrate a metadirectory component with their directory products. Most of the metadirectories available today require that the metadirectory internal data is stored in the same manufacturer's directory product.

Metadirectory is a directory-enabled application, where the application is the management of heterogeneous directory systems, and the role of the

associated directory is primarily to store configuration and mapping data, but not necessarily data that is vital for the role of an enterprise directory. The benefits of an effective metadirectory solution is that it is an enabler of a single point of administration and a single sign-on, and may also be used to provide:

- Lower administrative costs by acting as the source for consolidated directory information
- An authoritative source for public key certificates
- A central point for policy information
- Efficient access directory information for new Web-based applications

Because metadirectory is an integration tool, the directory product associated with the metadirectory solution may also be used as a company's enterprise directory, or it may exist as a stand-alone application.

1.4.1 Metadirectory systems

As shown in Figure 14, a metadirectory system may consolidate either:

- A superset of data derived from multiple different repositories, with different attributes managed from different systems; or
- A subset of data, maintaining the enterprise's common namespace alongside namespace mappings to all the connected systems, as well as one or two other key attributes, such as e-mail and public key

Either approach is valid, and they depend very much on a company's strategy for directory use and application development.

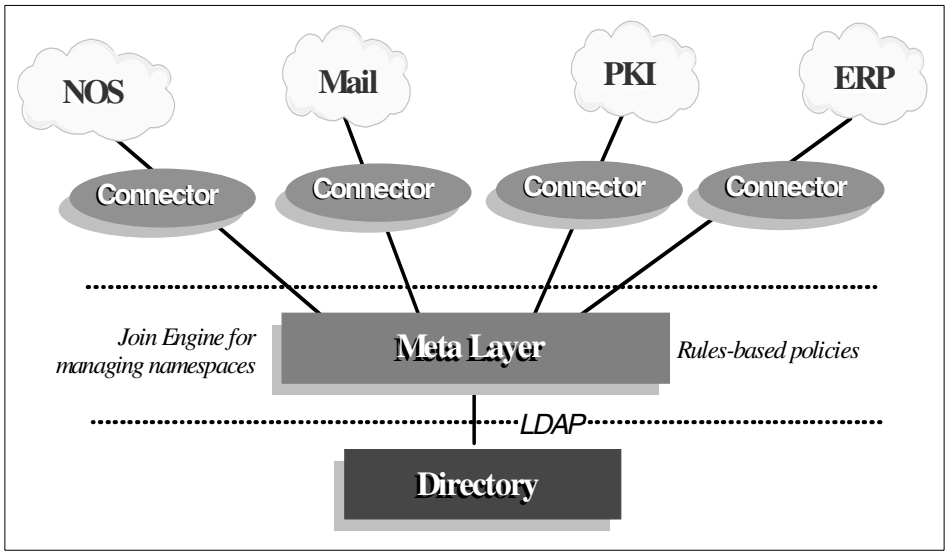

Figure 14. A metadirectory architecture

1.4.2 Metadirectory product architecture

As shown in Figure 15, typical metadirectory implementations include:

- Connectors: synchronization agents to extract and/or change information in source and destination repositories. Most products have connectors for relational databases, NOS, e-mail and other application directories.
- Join Engine: a central management process to direct the agents and transform data. This engine implements business rules describing how data is accessed and by whom. Sometimes, this component is also called *integration engine*.
- Broker: a process for distributing changes back to the directories. This process is considered part of the join engine, on some implementations.
- Optionally, a central LDAP-accessible directory.

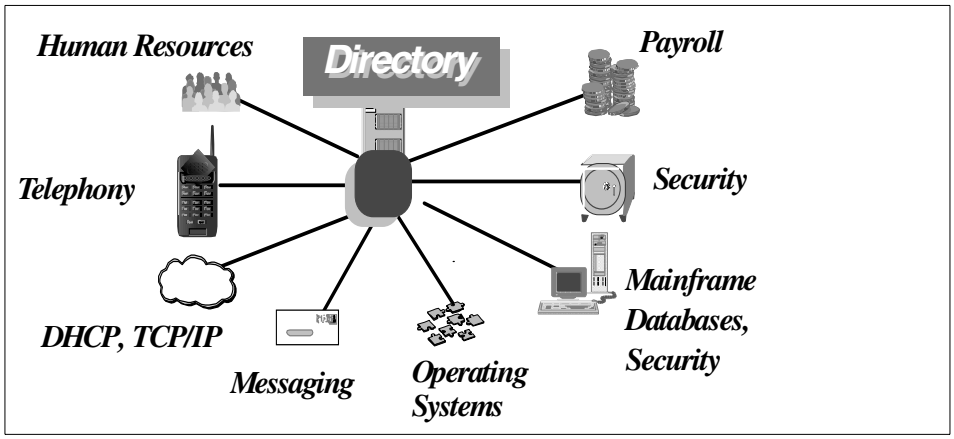

Figure 15. A metadirectory

Most implementations also include:

- A scheduler to manage the operational flow of the join engine and connectors.
- Scripting tools, either proprietary or based on standard scripting languages, to start auxiliary processes and provide programmatic control of both the join engine and connectors.
- A graphical or Web user interface to manage objects, configure directory views and access results.

1.5 The IBM SecureWay Directory and Client SDK

The IBM SecureWay Directory implements the IETF LDAP V3 specifications. It also includes enhancements added by IBM in functional and performance areas. This version uses the IBM DB2 as the backing store to provide per-LDAP operation transaction integrity, high-performance operations, and online backup and restore capability. It also interoperates with the IETF LDAP V2 based clients. Other major features include:

- A graphical directory management tool – The IBM SecureWay Directory Management Tool (DMT) allows an administrator to easily browse the directory and add or edit objects, such as object classes, entries and attributes.
- A dynamically extensible directory schema – This means that administrators can define new object classes and attributes 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 rebooting the directory server. Since the schema itself is part of the directory, schema update operations are done through standard LDAP APIs. The major functions provided by LDAP V3 dynamic extensible schema are:

  - Queryable schema information through LDAP APIs
  - Dynamic schema changes through LDAP APIs
  - Server rootDSE
  - Dynamic configuration changes using LDAP APIs

- The Directory-Enabled Network (DEN) schema – The DEN specification defines a standard schema for storing persistent state and an information model for describing the relationships among objects representing users, applications, network elements and networking services. More about DEN may be found in Chapter 7, “Future” on page 143. Also, complete DEN information can be found in the Distributed Management Task Force Web site (`http://www.dmtf.org`).
- Subclassing – Supports object inheritance for object class definitions and attribute definitions. A new object class can be defined using parent classes (multiple inheritance) and the addition or change of attributes. Schema update operations are checked against the schema class hierarchy for consistency before being processed and committed.
- UTF-8 (Universal Character Set Transformation Format) – An IBM SecureWay Directory server provides translation of data between multiple languages and UTF-8. It allows users to store, retrieve and manage information from/to a native language code page.
- Simple Authentication and Security Layer (SASL) – This support provides for additional authentication mechanisms. SSL provides encryption of data and authentication using X.509 V3 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, thereby improving the performance and reliability of the directory service.
- Referrals – Support for LDAP referrals allows directories to be distributed across multiple LDAP servers where a single server may only contain a subset of the whole directory data.
- Access control model – A powerful, easy-to-manage access control model is supported through ACLs (access control lists). Attribute-level access control is also supported.

- Support for DNS – An LDAP server can be located through DNS (Domain Name System) by publishing the Directory server address via a Service Resource record in DNS, and clients can find the LDAP server knowing only a symbolic name.
- Support for server plug-ins – An extensible server architecture that allows implementers to write server plug-ins that carry out additional functions for the server to perform. A plug-in is a dynamic link library (or shared object) that can be included in the server's address space dynamically. They follow the plug-in APIs published by Netscape, and directory plug-ins are compatible with the Netscape iPlanet Directory Server.
- Security auditing – A configurable security auditing service is provided that will log LDAP activity including information such as timestamps, client IP addresses, the Bind DN associated with the connection, etc. This service is provided as a server plug-in, so that administrators have the option of providing their own plug-in to direct the audit stream to OS-specific or application-specific audit facilities.
- Kerberos V5 authentication – Client and server-side SASL plug-ins are provided supporting the GSSAPI SASL mechanism. The GSSAPI mechanism will enable kerberos authentication from both SecureWay and Windows 2000 LDAP clients.
- Change Log – The directory may be configured to maintain a change log. This change log records information about updates to the directory. This information is especially useful for use by meta-directory products or for applications that cache data (they can easily check to see what updates have affected their cached data).
- Event Notification – Allows clients to register to be notified of changes to entries or portions of the DIT. The registration specifies what types of updates (add, delete, modify, moddn, all) the client wants to be notified of.
- Limited Transactional support – The server provides for some application control over the scope of a directory update transaction. By default, each individual LDAP operation is treated as a separate transaction with the directory, but LDAP extended operations are provided to group a sequence of updates into a single transaction. The transaction is limited to a single connection to a single LDAP server. An LDAP extended operation is provided to start a transaction. Following the StartTransaction, the application may submit some reasonable number of asynchronous update operations, followed by an extended operation to end the transaction (either committing the set of updates or rolling them all back). If any individual update within the transaction fails, the entire transaction will be rolled back. The updates are not committed to the database until all

updates have succeeded, and an EndTransaction-Commit has been received.

The LDAP V3 RFCs implemented in the IBM SecureWay Directory are RFC 2251, RFC 2252, RFC 2253, RFC 2254, RFC 2255, and RFC 2256. You can find more information about these RFCs in Appendix D, “Standards” on page 175.

Administration and configuration for the IBM SecureWay Directory is provided by a Web browser-based GUI. The administration and configuration functions allow the administrator to:

- Perform the initial setup of the directory
- Change configuration parameters and options
- Manage the daily operations of the directory

The IBM SecureWay Directory is available on IBM AIX, Microsoft Windows NT/Intel, and Sun Solaris platforms, and it is translated into 10 languages including English, French, German, Japanese, Simplified Chinese, Traditional Chinese, Korean, Italian, Spanish and Brazilian Portuguese. Catalan is also supported on AIX.

The IBM SecureWay Directory is also available for OS/390 and OS/400. They are integrated in the OS/390 Security Server software and OS/400 Operating System, respectively. The OS/390 and OS/400 implementations are currently on LDAP V3 level.

In addition to the IBM SecureWay Directory, which incorporates an LDAP server, IBM also provides a client development kit that provides the necessary APIs and libraries for development of directory-enabled applications. The IBM SecureWay Directory Client SDK also contains the directory management utilities.

The IBM SecureWay Directory Client SDK is available for IBM AIX, Microsoft Windows NT/95/98, Sun Solaris and HP-UX. Please check the IBM SecureWay Directory Web site at `http://www.ibm.com/software/enetwork/directory` for the latest information about products, supported platforms, and availability.

1.6 Lotus Domino R5

Lotus Domino supports a variety of business applications including an LDAP-based directory service. The Domino directory is designed to serve as a key enabling technology for a directory-enabled infrastructure. To leverage

the inherent value of this potentially rich and useful store of corporate information beyond e-mail addresses and certificates, Domino/Notes customers can exploit this directory architecture, which they already have in place.

Today's Domino Directory can be part of a general-purpose directory infrastructure for the enterprise and for multi-enterprise extranets. On the other hand, in a heterogeneous networking environment with other directory systems, the Domino Directory can also serve as the integration point for directory synchronization, administration, and authentication.

Directory features in Domino R5 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 (for example Active Directory and Four11). This allows for developing a distributed user authentication mechanism, making it easier to achieve a single sign-on environment.
- 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.
- An open architecture that can easily incorporate support for emerging standards.

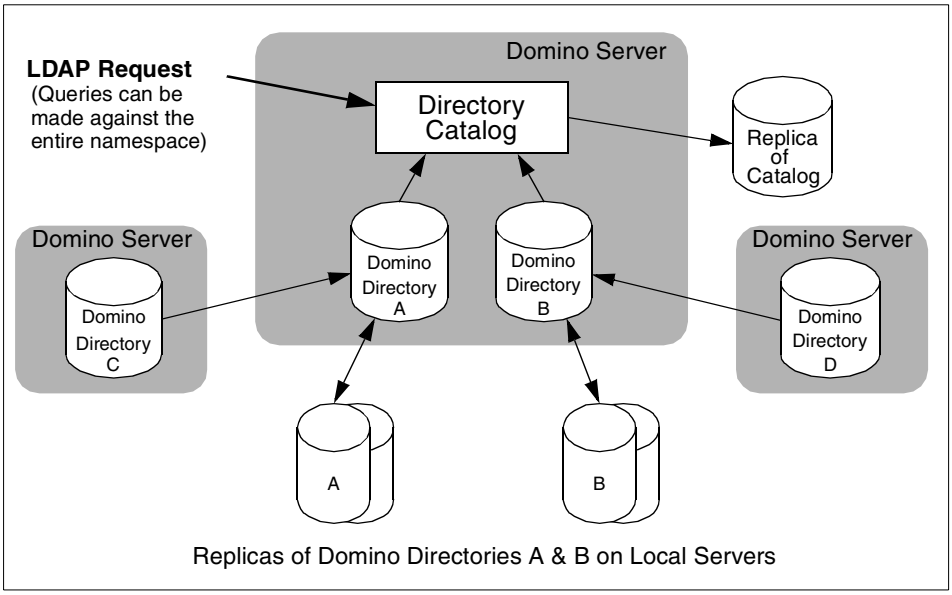

Figure 16. The Domino 5.0 directory architecture

Lotus Domino R5 is available for all the IBM platforms (AIX, OS/2, OS/390, and OS/400), Windows NT, Solaris and HP-UX platforms.

1.7 Microsoft Active Directory

Microsoft Windows 2000 operating system includes the Active Directory, an LDAP Version 3-compliant directory service.

Active Directory follows RFC 2247 conventions for naming contexts, meaning that its naming context suffixes map directly to the DNS tree.

Some Active Directory features include:

- Centralized management
- Group policies
- A global catalog
- IntelliMirror desktop management and automated software distribution
- Multi-master replication
- Kerberos authentication
- Smart card support

- PKI/x.509 certificate support
- Attribute-level security

(For more information on Windows NT and Windows 2000 domains, please refer to the respective product's documentation.)

1.7.1 Naming contexts

In Active Directory, naming contexts are the primary unit of replication. The Active Directory always has at least three naming contexts:

- The schema
- The configuration, which holds the replication topology and metadata
- User naming contexts, which are the body of the directory and hold the actual directory objects.

1.7.2 Logical elements

1. Domains

   The Active Directory is made up of at least one domain. Microsoft defines a *domain* as "a logical grouping of network servers and other computers that share common security and user account information".

   The main implications of this concept are that:

   - A domain can span multiple physical locations. Since it is a *logical* grouping, it bears no relation to the underlying physical network.
   - A domain is a completely independent unit. If all other domains that have relationships to it were to disappear, all objects in the domain would be unaltered in the functionality.
   - A domain has its own security. This means every element of a domain must comply with its security, and cannot be under the influence of the security of other domains. It may have security relationships to other domains, however.

2. Trees

   One or more domains that share a contiguous namespace are called a *tree*. The domains in the tree are linked, from a security standpoint, by implicit Kerberos trusts. These Kerberos trusts are transitive and hierarchical.

   A tree is usually called by the name of the domain at its root.

3. Forests

When you have two or more trees with non-contiguous namespaces, you have a *forest*.

Forests don't have actual names, because they are just a set of cross references and Kerberos trusts, but it is usual to call them by the name of the tree that is at the root of the forest (from a Kerberos security hierarchy standpoint).

Both trees and forests share a common schema, configuration, and global catalog.

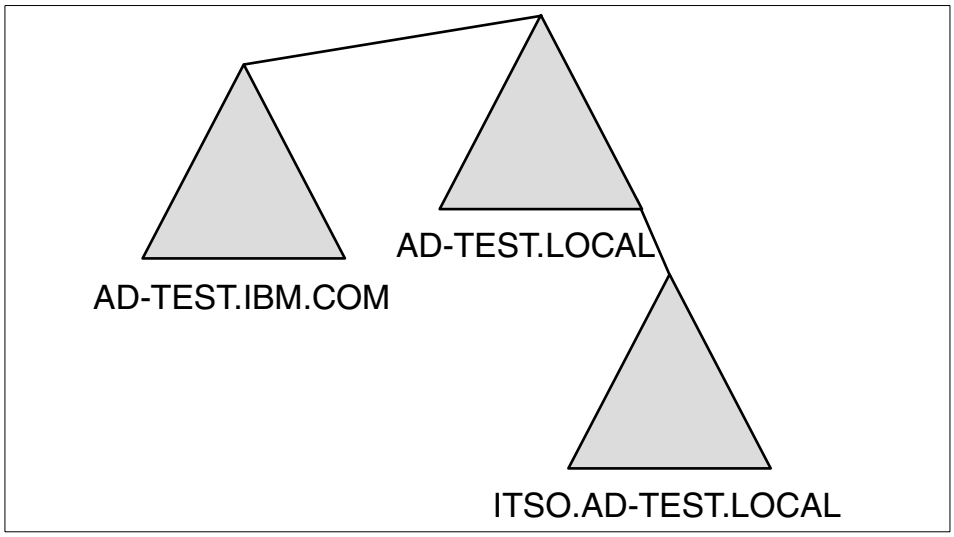

Figure 17. A forest with three domains on two trees

A note about trees and forests

You should try not to create multi-domain trees, except if your company's policies dictate a completely separate and independent security, such as user rights and password policies. When you create another domain, you lose the capability of sharing policies and thus avoiding rework.

You should never create a forest unless you have a very good reason to do so. An example of a situation where you would have to create a forest would be a company merger, where you have two separate namespaces to deal with.

1.7.3 Physical elements: sites and domain controllers

The main physical elements of Active Directory are *sites* and *domain controllers*.

A *site* represents a physical location, usually with good network connectivity. A site is composed of one or more IP subnets. The main purpose of the site is to separate the logical, administrative aspects of the network from the physical, performance-related ones, so that network administrators don't have to worry about the physical network when designing the logical domain.

It is a very simple mechanism, because the DNS server will always provide the client machine with a list of IP addresses to all domain controllers, and, since the client already knows its own address, it will always be able to find domain controllers in its own site (provided that they exist, of course). The client will always try those servers first. Replication traffic between domain controllers will also have different behaviors regarding scheduling and network bandwidth used, depending on how the domain controllers map to the sites.

A *domain controller* is a computer that holds a complete replica of the Active Directory subtree that represents its domain. The number and location of the domain controllers in a forest and at a given site will greatly influence the performance and reliability of the network, as well as those of any directory-enabled applications that may be in use.

1.7.4 Architecture

1.7.4.1 Data model

The data model of the Active Directory is derived from the X.500 model. The three main concepts are the schema, classes and objects.

- *The schema* represents the universe of all possible classes, attributes and objects. In the Active Directory implementation, the schema itself is stored as a set of classes in the database (as opposed to a flat text file, loaded at initialization time). This has several advantages: it allows for easier access from applications, it allows for dynamic changes to the schema without the need to stop and restart the directory service, and it allows for uniform security, by using the same ACLs used for objects.
- For each *class* there may be *mandatory attributes*, *additional attributes*, and a list of possible *parents* (classes). Each class must have an *object identifier* (OID) that will identify it unambiguously. Each individual or organization that wishes to extend the schema should obtain a *root OID* from an issuing authority. The OIDs are represented in dotted decimal form, so, for example, every LDAP standard class starts with 2.5.6, which

means it is jointly defined by ISO and the ITU (2), as part of the Directory Standard (5) and it is a class (6). The individual or organization can then manage further branches of its root OID to suit his/her/its needs. Please see Appendix E, “X.500 object identifiers (OIDs)” on page 179 to learn more about OIDs, the issuing authorities, and their uses. The `http://www.alvestrand.no/objectid/` Web site is also a very good source of information on this subject.

- The *objects* are actual instances of the classes. As an analogy, you may think of the classes as tables in a database, and the objects as the data in these tables.

1.7.4.2 Naming conventions

Some of the naming conventions used in the Active Directory implementation are:

- Distinguished Name (DN): it represents the full object name, from the root, such as “CN=Bernard Freund, CN=Users, DC=ECPWEB, DC=COM, DC=BR” and it uses the format defined on RFC 2247 (for more information on this RFC, see Appendix D, “Standards” on page 175). The DN is guaranteed to be unique across the forest.
- Relative Distinguished Name (RDN): it represents the portion of the DN that is an attribute of the object itself. In the preceding example, the RDN of the user is “CN=Bernard Freund” and the RDN of the “Users” container is “CN=Users”.
- User Principal Name (UPN): it is an easy-to-remember alias for the user, in the form <username>@<DNS domain name>, very similar to the user’s Internet e-mail address. It is guaranteed to be unique, but only *security principals* (that means objects that can be used on ACLs: users and security groups - see 1.7.4.3, “Administration” on page 42 for more information on ACLs) have UPNs.
- Globally Unique Identifier (GUID): this is an identifier that is statistically guaranteed to be unique for all objects in the database. All objects are identified by their GUIDs for as long as they exist. The GUID is created when the object is created. It never changes, and, when the object is destroyed, its GUID is lost forever.
- The Security Identifier (SID) is a unique identifier assigned to security principals (users and security groups). The object's SID is actually composed of two parts: the domain SID, common to all objects in a domain, and the Relative Identifier (RID), that is unique within a domain. The SID is used on all ACLs for authorization (permissions) purposes. The SID's behavior is very similar to the GUID's.

Note

Some operations on Active Directory require that all directories involved comply to RFC 2247 “Using Domains in LDAP/X.500”. For more information on this and other RFCs, please see Appendix D, “Standards” on page 175.

1.7.4.3 Administration

All authorization and authentication must come from a higher authority, usually Kerberos. Each object in the Active Directory, including the schema objects, has a set of actions that can be performed on it, called an *access control list* (ACL). The ACL contains sets of GUIDs and the rights associated with these, so it basically says who can do what on the object.

The ACLs are inherited down a subtree, so unless the inheritance is explicitly blocked, the ACLs will propagate down from containers to subcontainers and leaf objects, providing for a somewhat easy form of delegation, by assigning ACLs to OUs.

Finally, the *Directory Service Agent* (DSA) manages physical storage of the objects and classes in the directory database, providing for client isolation.

1.7.5 The role of DNS

The Active Directory is heavily dependent on the DNS service for all its operations. The three main roles that DNS performs are:

- Name resolution: DNS maps host names to IP addresses, without the need for a WINS service (please note that for older Windows clients to access Windows 2000, there will still be a need for WINS).
- Namespace definition: the Active Directory namespace maps to the DNS namespace, simplifying the namespace design and use.
- Locating physical components of the DNS service by using SRV records. Thus, Windows 2000 clients and servers may find site and domain controller information quite easily (see 1.7.3, “Physical elements: sites and domain controllers” on page 40 for more information on this subject).

1.7.6 Special roles

Although most operations on the Active Directory are *multi-master* operations (meaning that you can read or write to any domain controller indiscriminately), a few operations need a “focal point” for greater consistency or performance reasons. These have been divided according to their scope:

- Forest-wide roles (at least one per forest):
  - Global catalog: The global catalog stores a subset of the attributes of all objects in the forest. Those are usually the attributes most used on searches. The careful placement of global catalog servers greatly improves performance, especially if you are using directory-enabled applications. You should have at least one global catalog server on each physical site.
  - Schema master: The schema master is the focal point for making changes to the schema. Since this is a very infrequent and sensitive operation, the Active Directory allows making changes to the schema only through the schema master. There can be only one schema master per forest.
  - Domain naming master: The domain naming master is responsible for controlling the addition or removal of domains in the forest. There can be only one domain naming master per forest.
- Domain-wide roles (at least one per domain):
  - RID master: RID (after domain SID).
  - RID master: responsible for allocating RIDs from the domain's RID pool (RID is the part of the object's SID located after the domain SID).
  - PDC Emulator: Services non-Windows 2000 clients for logons and password changes. Also preferential replication of passwords on Windows 2000.
  - Infrastructure Master: Updates group-to-user references when membership changes (user display name would be an example).

Chapter 2. Scenario1: Integrating SecureWay with Active Directory

The following scenario is of two fictional companies that have recently merged. One of the companies uses the IBM SecureWay Directory as its directory. The second company has a Windows 2000 domain, and it uses solely the Active Directory. In this scenario we will explore referral functionality by having an LDAP client search for data stored in a directory (the IBM SecureWay Directory) by connecting to another directory (Active Directory) and vice versa.

2.1 About our test environment

For this scenario, we have set up a very simple computing environment, consisting of one Windows NT 4.0 server running the SecureWay product and one Windows 2000 domain controller, as shown in Table 2:

Table 2. Specifications for the first scenario

Machine DNS Name	**LDAP1.AD-TEST.IBM.COM**	**AD-TOP.AD-TEST.IBM.COM**
Operating system	Windows NT Server 4.0 SP6a	Windows 2000 Advanced Server SP1
Domain	(none)	ad-test.ibm.com
Domain role	N/A	domain controller
Additional software	IBM SecureWay Directory V3.2	(none)
	DB2 UDB V6.1.0.21	
	Java Developer Kit (JDK) V1.1.8	
Additional Windows components	Microsoft IIS V3.0	Microsoft IIS V5.0
		Active Directory

The scenario environment is illustrated in Figure 18.

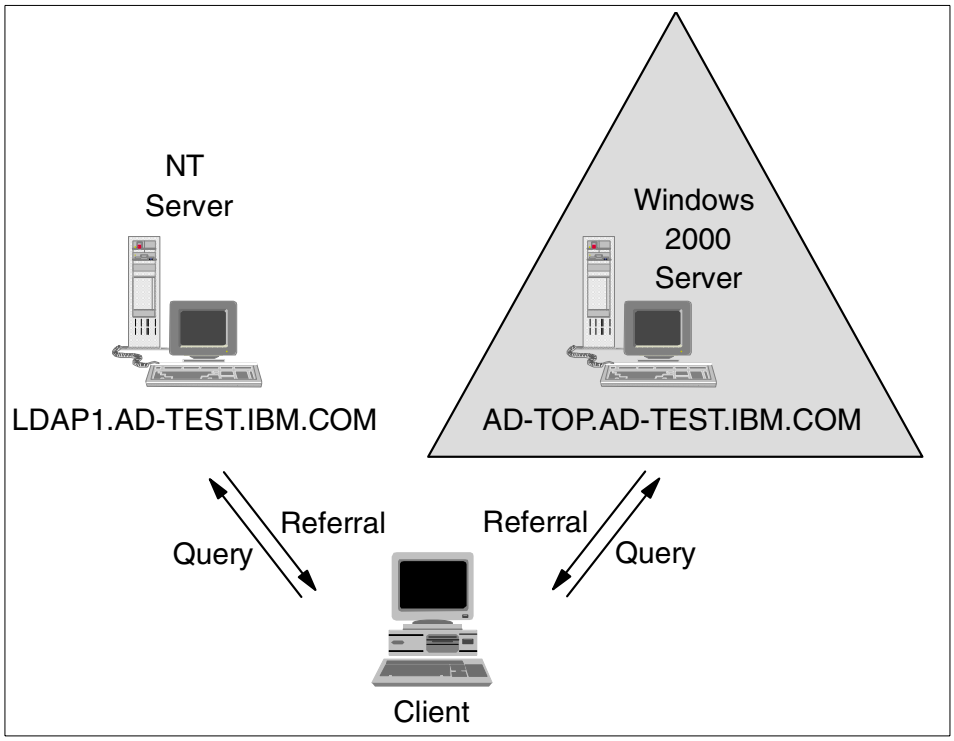

Figure 18. First scenario environment

For the sake of simplicity, AD-TOP is the primary DNS server for all forward and reverse DNS zones, and all zones are standard zones.

2.2 How we tested

Again, for this scenario we wanted to keep things simple, and deliver just the basic interoperability before delving into more complicated issues.

Therefore, we decided to use this scenario to introduce some of the basics of creating referrals and using them to leverage existing directories. The next chapters will build upon these to add functionality and complexity, in a step-by-step approach towards directory integration.

2.3 Configuring SecureWay

For instructions on installation and configuration of SecureWay refer to the *LDAP Implementation Cookbook*, SG24-5110. Once all the software is

installed, SecureWay needs to be customized. The following steps outline what was done.

First a domain suffix of ‘o=IBM, c=US’ was added, through the IBM SecureWay Directory Server Administration interface.

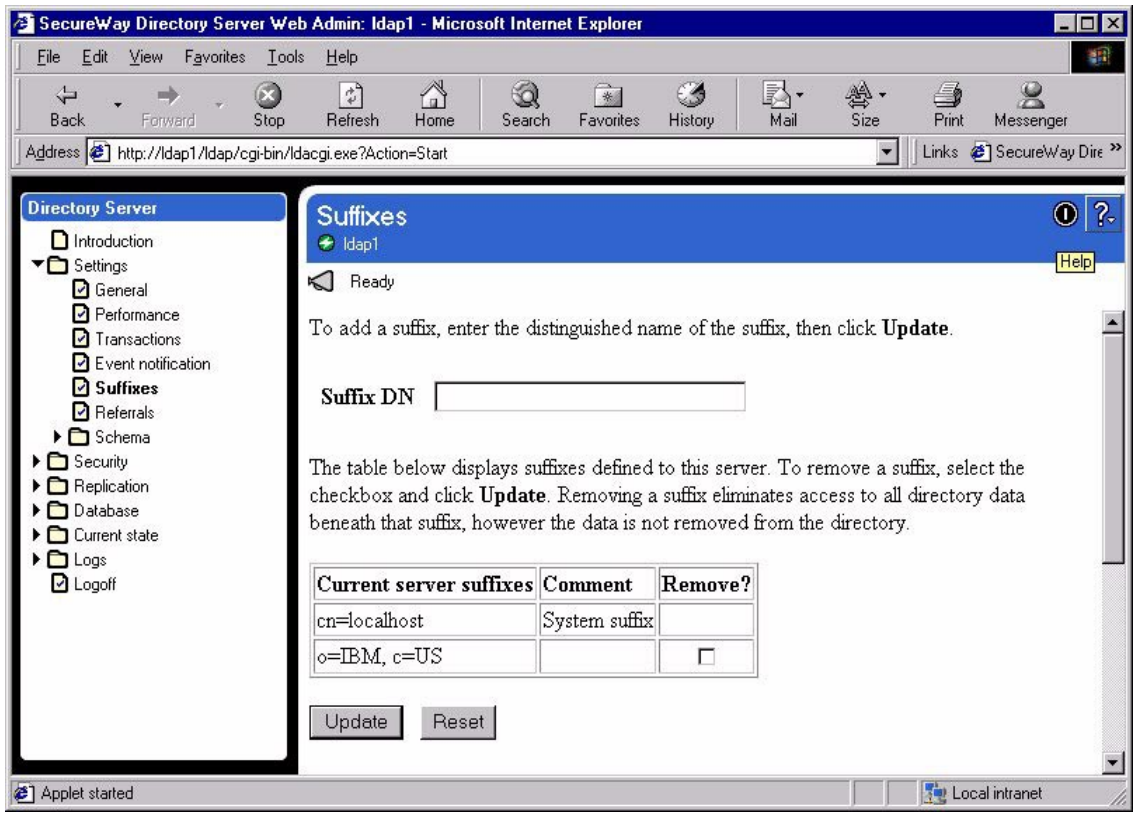

Figure 19. SecureWay - adding a suffix

Then a sample LDIF file was imported to create a sample directory, through the IBM SecureWay Directory Server Administration interface.

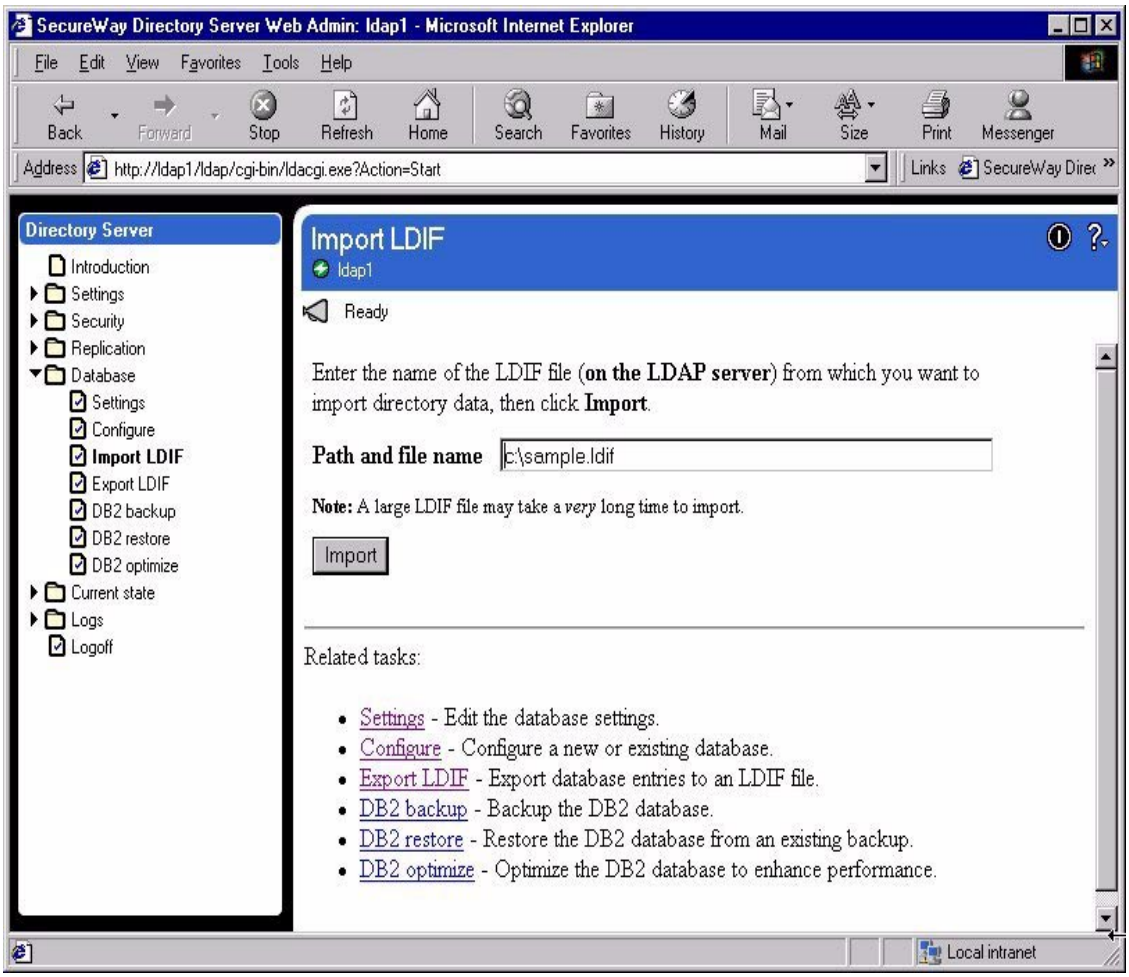

Figure 20. SecureWay- importing an LDIF (1)

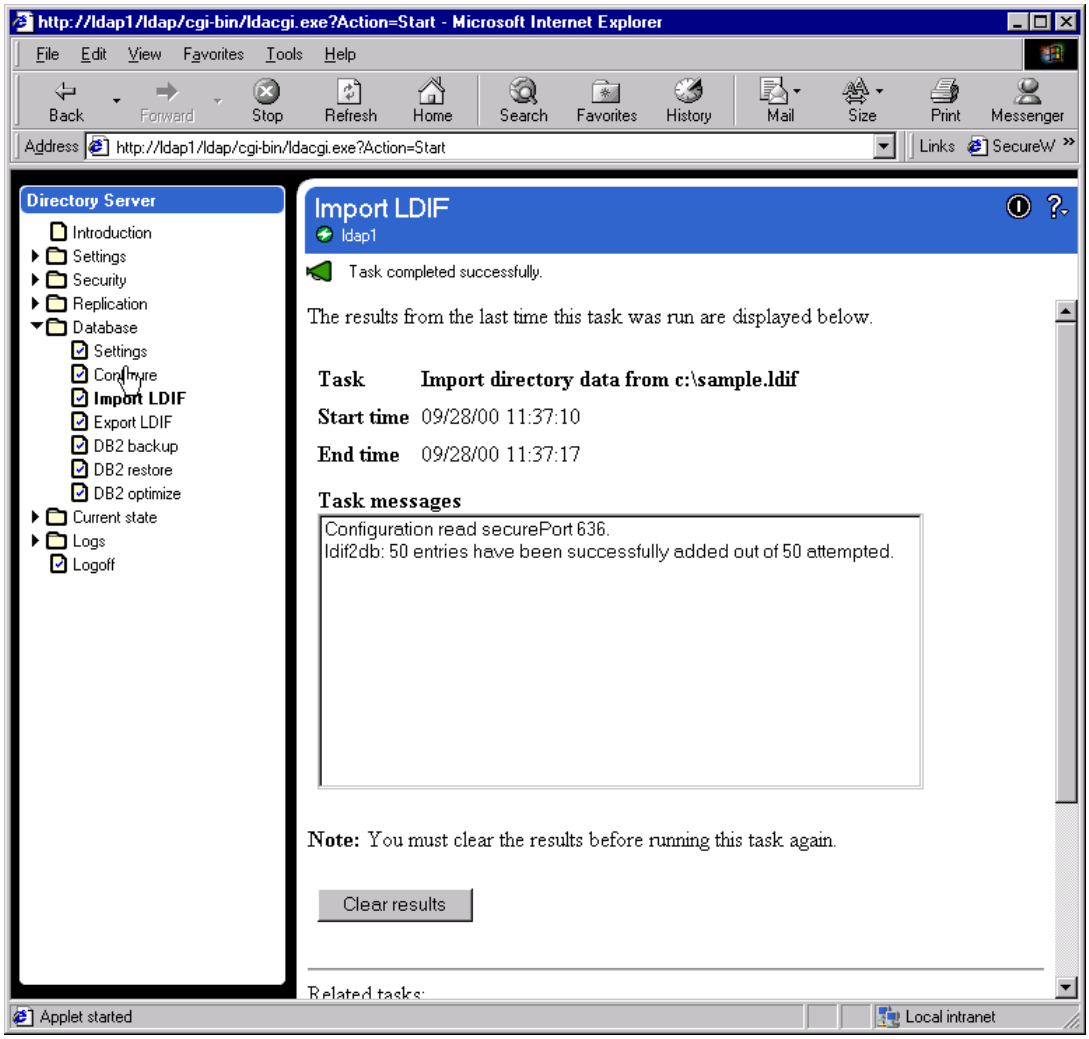

Figure 21. SecureWay- Importing an LDIF (2)

2.4 Creating a referral in SecureWay

To set up SecureWay for referrals, enter the Referral URL in the format `ldap://hostname:port` and a DN of the referral object. This is done through the IBM SecureWay Directory Server Administration interface.

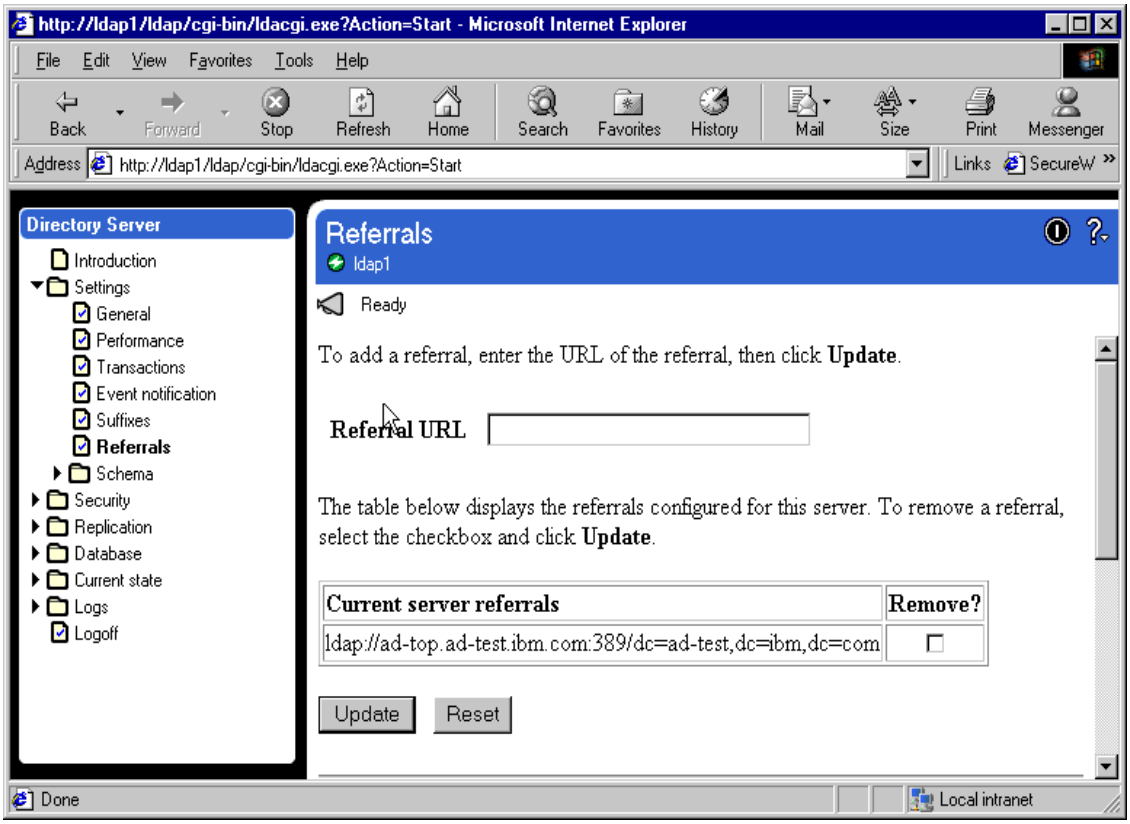

Figure 22. SecureWay - adding a referral

Note

For referrals to work you need to ensure that there is sufficient permission to access the directory tree. Directory tree security may be updated through the IBM SecureWay Directory Management Tool.

2.5 Creating a referral in Active Directory

To configure the Active Directory to do referrals to external naming contexts, we have used the ADSI Edit Microsoft Management Console (MMC) snap-in. Instructions for installing this and other Windows 2000 tools used throughout this chapter may be found in Appendix B, “Client tools” on page 155.

To configure the Active Directory for referrals, we have used the ADSI Edit console (click **Start** -> **Programs** -> **Windows 2000 Support Tools** -> **Tools** -> **ADSI Edit**).

Note

Instructions for creating an empty console and adding the ADSI Edit snap-in to it, or adding the ADSI Edit snap-in to your own MMC console are provided in Appendix C, “Setting up MMC consoles” on page 161.

The next step in this process is connecting to the correct naming context in the Active Directory, in this case the configuration naming context.

To do this, we right-clicked **ADSI Edit** and chose the **Connect to...** option.

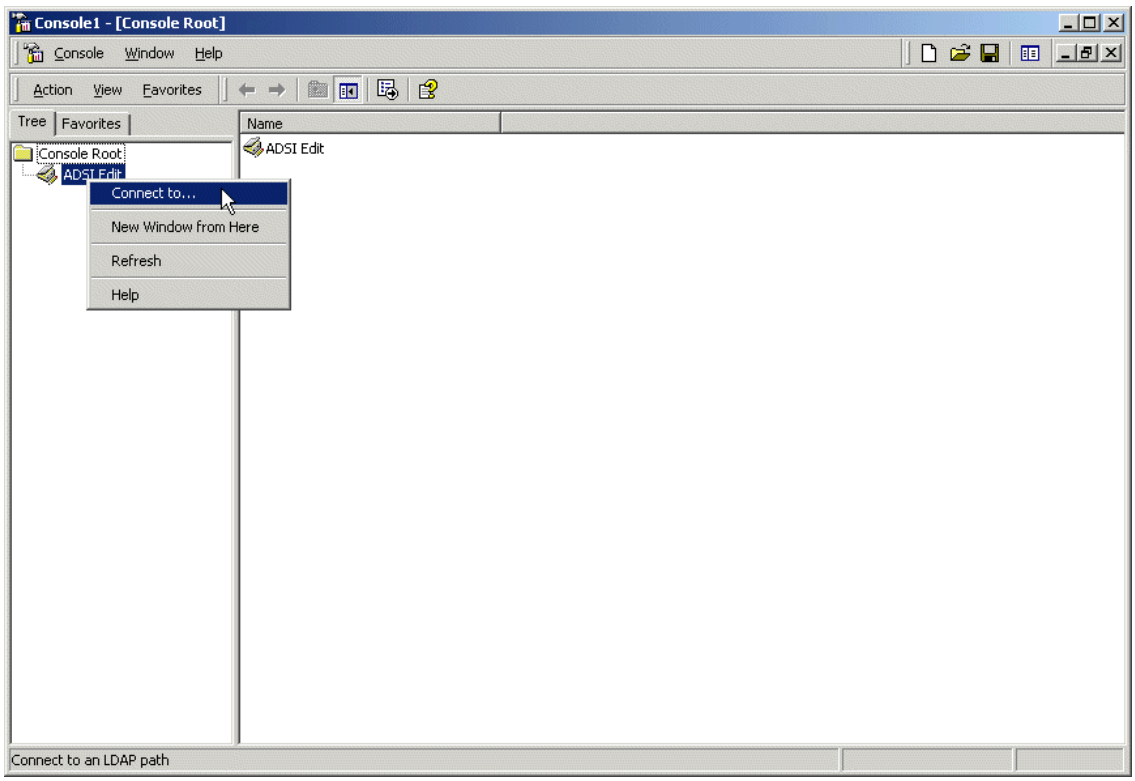

Figure 23. Connecting to a naming context

In the Connection dialog box, we changed the naming context to **Configuration Container**, and we clicked **OK** to close the dialog box. If you

want to choose a different domain or domain controller to connect to, you also can specify this in the same dialog box, by typing or selecting the domain or server name in the Select or type a domain or server field.

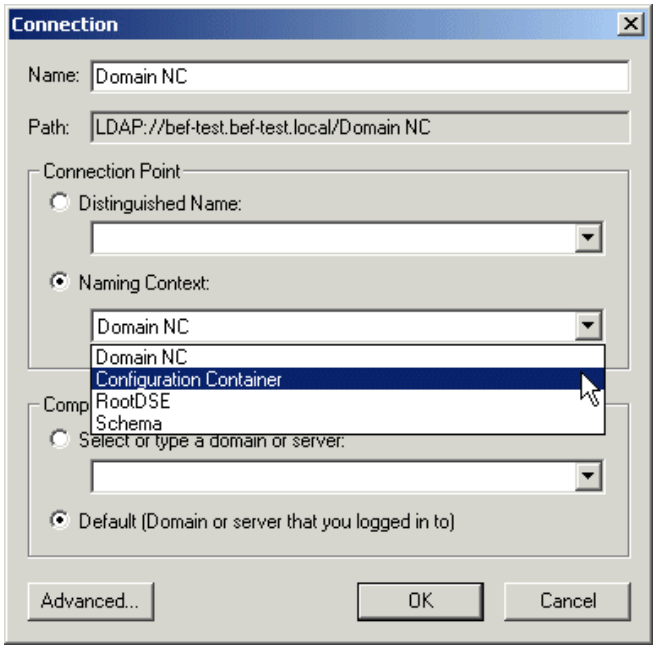

Figure 24. The Connection dialog box

Then we were connected to the correct naming context (Figure 25).

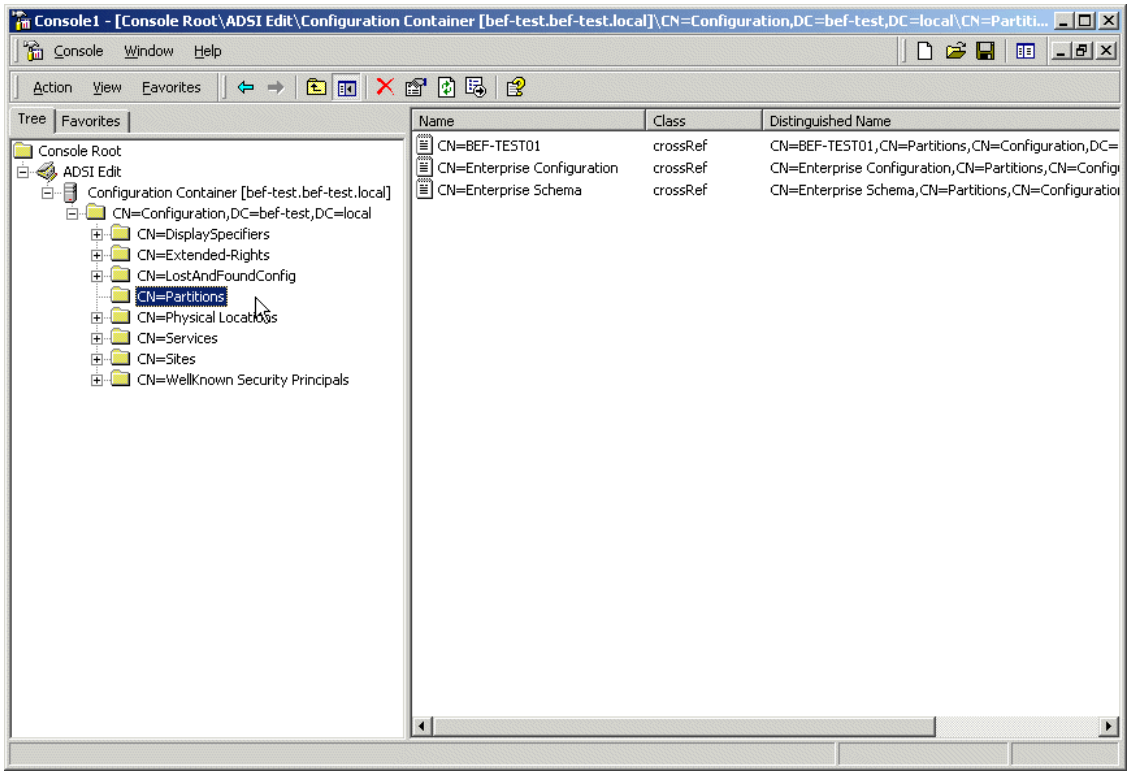

Figure 25. The Partitions folder

The next step was to actually create the referral object. To do this, as you can see in Figure 26, we right-clicked the **CN=Partitions** folder, and we selected **New**, and then **Object...** to bring up the Create Object dialog box.

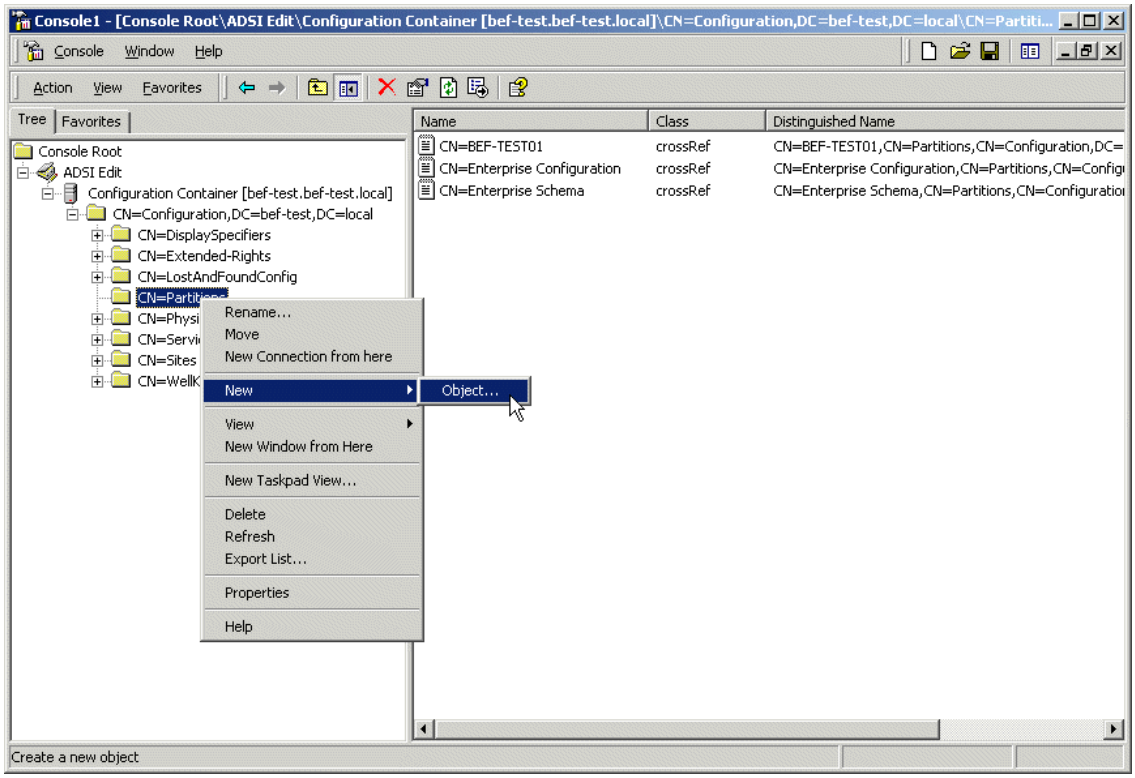

Figure 26. Adding an object

For the object class, the only choice available was crossRef (see Figure 27), which was exactly what we needed. So we clicked **Next**.

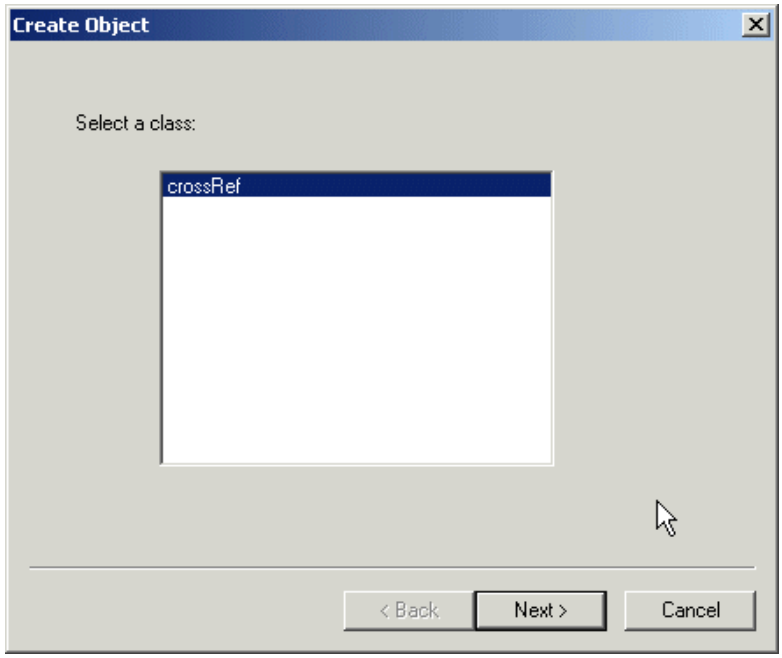

Figure 27. Adding an object: object class

The next window (Figure 28) asked us to name the object. This is the first mandatory attribute for the crossRef class: cn (common name). You can choose any name at this point. We have chosen **SecureWay Directory**.

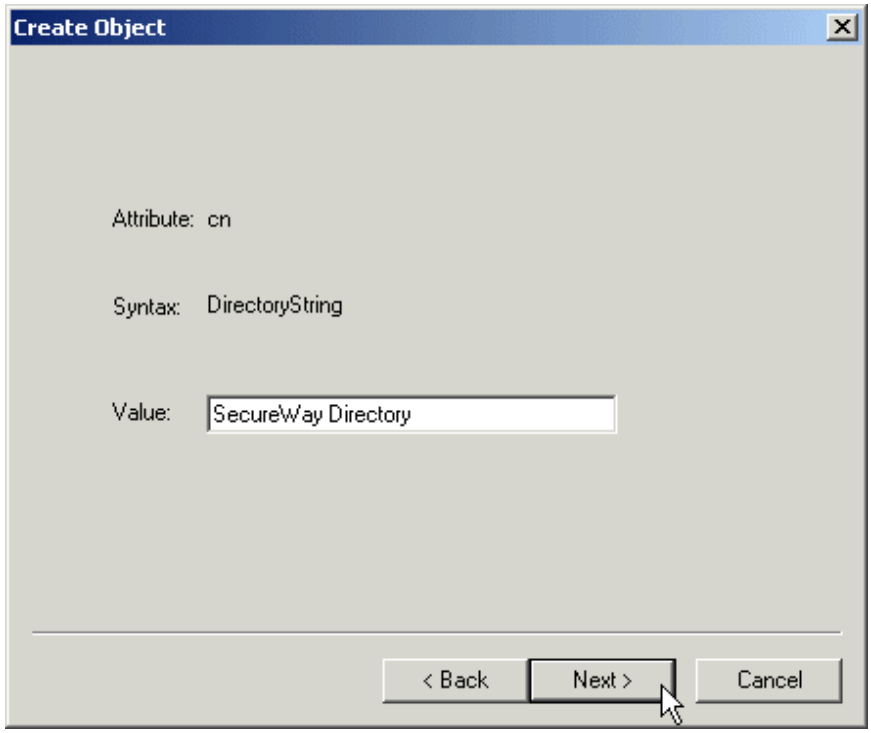

Figure 28. Adding an object: choosing a cn

The next mandatory attribute for this class (shown in Figure 29) is nCName (naming context name). This should be the naming context being referenced, in this case the naming context that we have created in the SecureWay Directory (see 2.3, “Configuring SecureWay” on page 46).

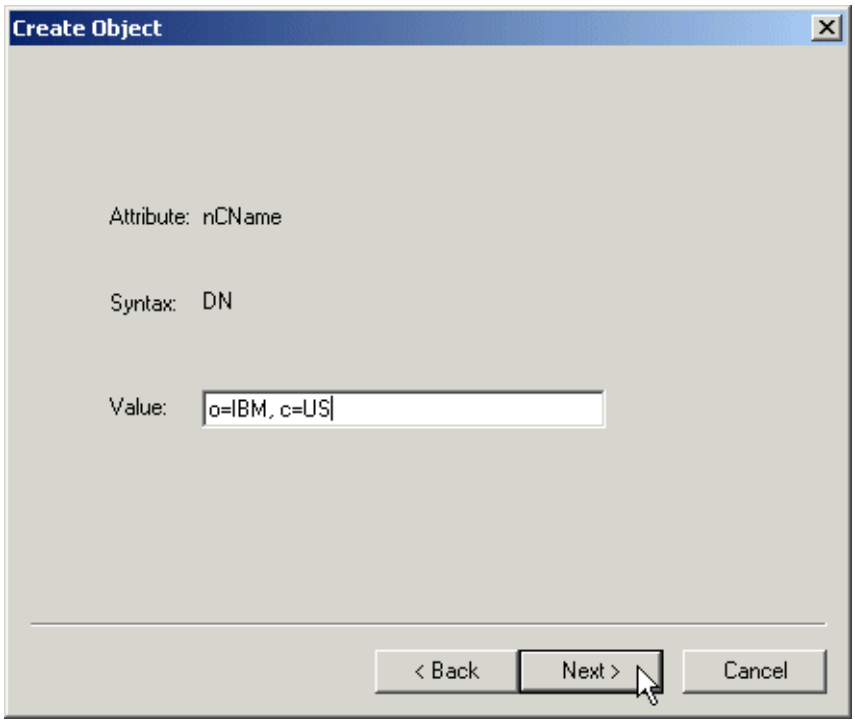

Figure 29. Adding an object: choosing the referenced naming context

The last mandatory attribute was dnsRoot (Figure 30). This should be the DNS name for the server that handles the naming context referenced in the previous window. This is also the value that will be returned to clients that query the active directory for information that belongs to the naming context mentioned earlier.

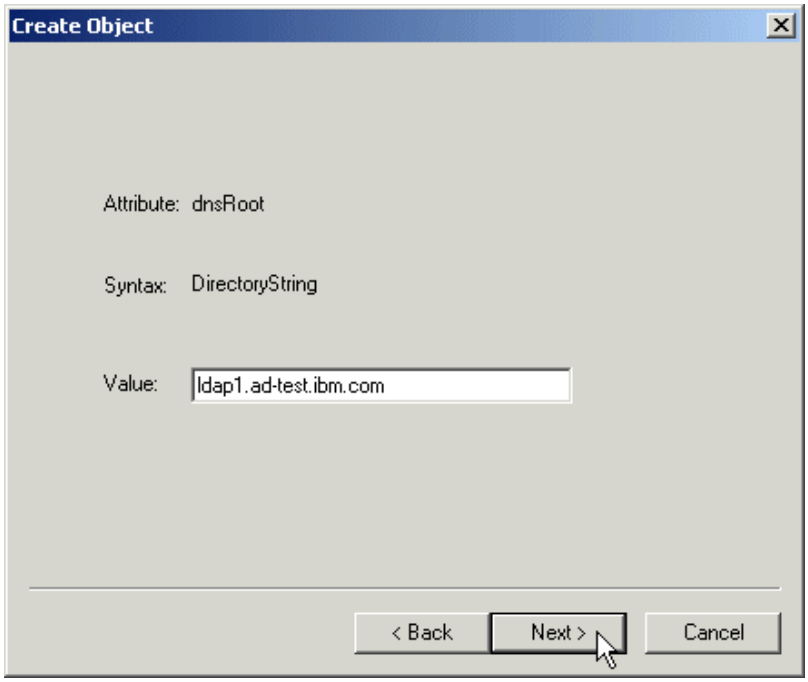

Figure 30. Adding an object: selecting the referred server

If you want to add any optional attributes to this class, such as an administrative description of this object, you can select the **More Attributes** button in the next window. We wanted to remain simple, so we chose **Finish**.

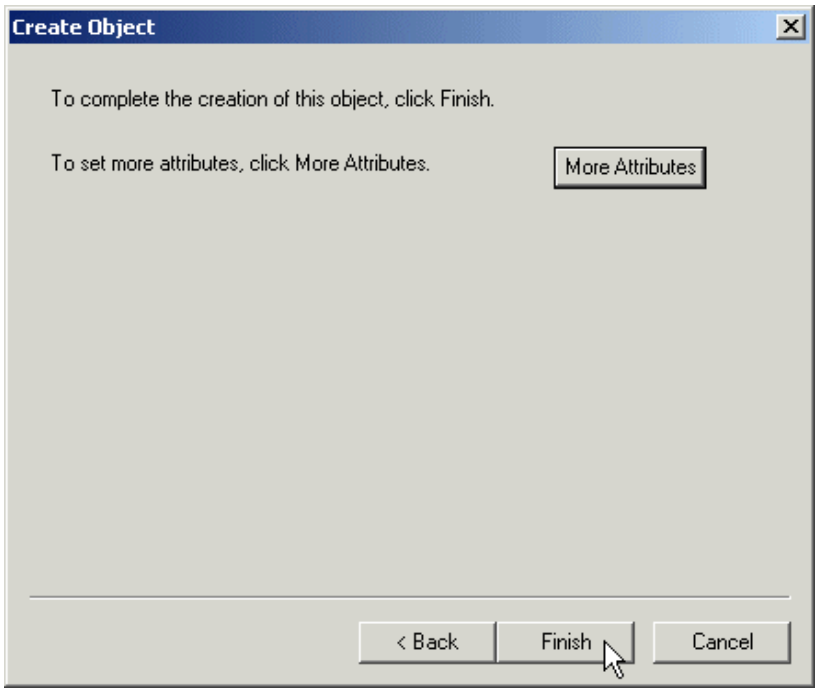

Figure 31. Adding an object: optional attributes

You can see the final result in Figure 32:

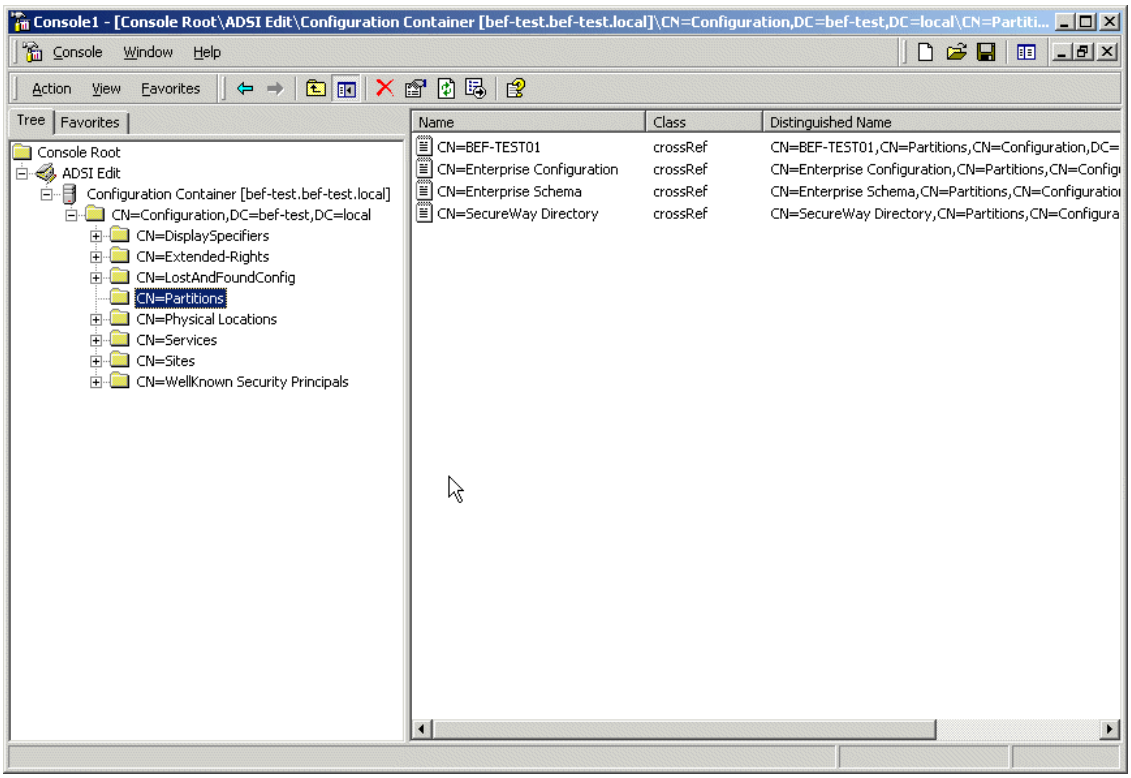

Figure 32. The final result

An important note on security

To make sure that both directories allow the operations required for the referrals to work properly, the access control rules should be applied consistently in both the Active Directory and the SecureWay Directory. These rules should allow the directory service agents from one directory to perform these operations in the other directory and vice versa. Such configuration goes beyond the scope of this book, but using the Kerberos support in SecureWay would be a good starting point.

2.6 Bringing it all together

Finally, we had both the Active Directory and the SecureWay Directory correctly configured. The final task for this scenario was to simulate a client doing a search for data in each directory service, while accessing the other.

We have analyzed six possible cases:

1. A client connects to an Active Directory server, and searches for data stored in the same server.
2. A client connects to an Active Directory server, searches for data stored in the SecureWay Directory and receives a referral.
3. A client connects to an Active Directory server, searches for data stored in the SecureWay Directory, receives a referral, and chases this referral.
4. A client connects to a SecureWay Directory server, and searches for data stored in the same server.
5. A client connects to the SecureWay Directory, searches for data stored in an Active Directory server and receives a referral.
6. A client connects to the SecureWay Directory, searches for data stored in an Active Directory server, receives a referral, and chases this referral.

2.6.1 Searching for data in the Active Directory

The simplest Active Directory case occurs when the client connects to a server and queries for data stored in the same server, such as a user or computer that belongs to the same domain as the server.

1. The first step is clicking **Start**, then **Run**, then pointing to the path to the LDP utility executable (for directions on how to install LDP and other utilities, please refer to Appendix B, “Client tools” on page 155).

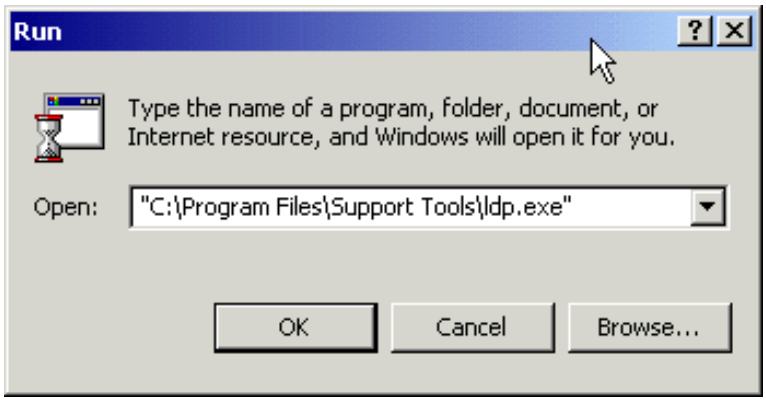

Figure 33. Starting the LDP utility

2. The LDP utility will start, as shown in Figure 34.

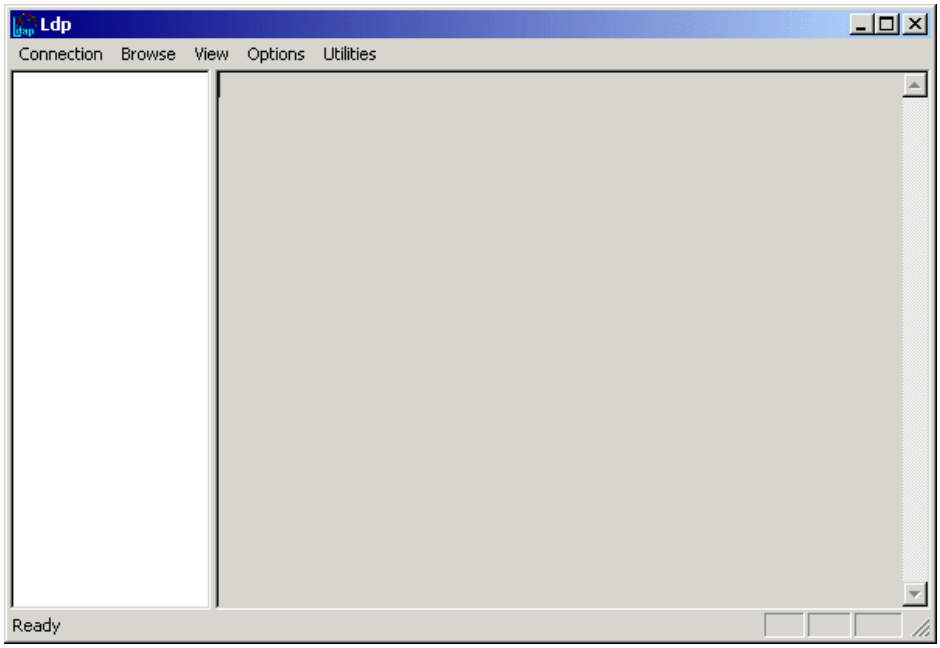

Figure 34. The LDP utility

3. The next step is to establish a session to the LDAP (Active Directory) server. To do this, click **Connection**, and then **Connect**, as shown in Figure 35.

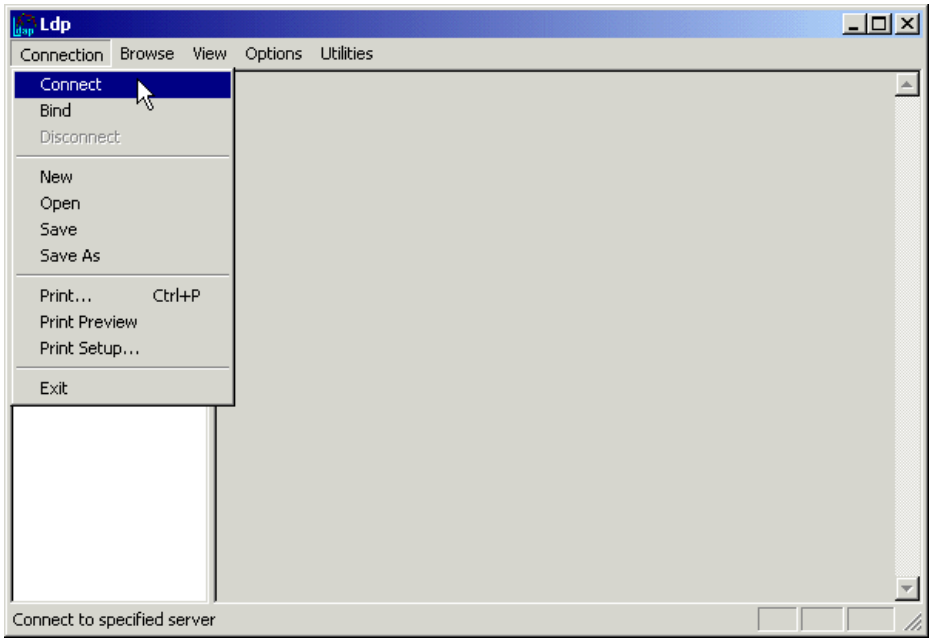

Figure 35. Establishing a session to the LDAP server

4. In the Connect dialog box (Figure 36), type the name or IP address and the port of the target LDAP server.

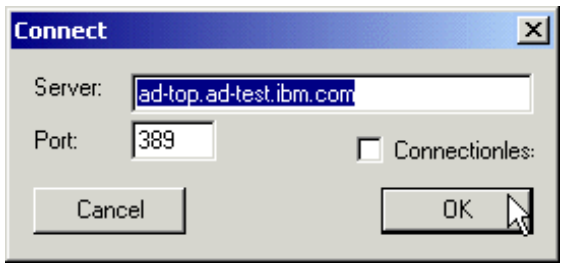

Figure 36. The Connect dialog box

5. The right-hand side of the window will show the standard LDAP V3 RootDSE, plus additional attributes. For a list of the RootDSE standard attributes, please see the LDAP V3 RFC and Appendix A, “The RootDSE” on page 153. For a list of relevant LDAP RFCs, please see Appendix D, “Standards” on page 175. You can see part of the Active Directory RootDSE in Figure 37.

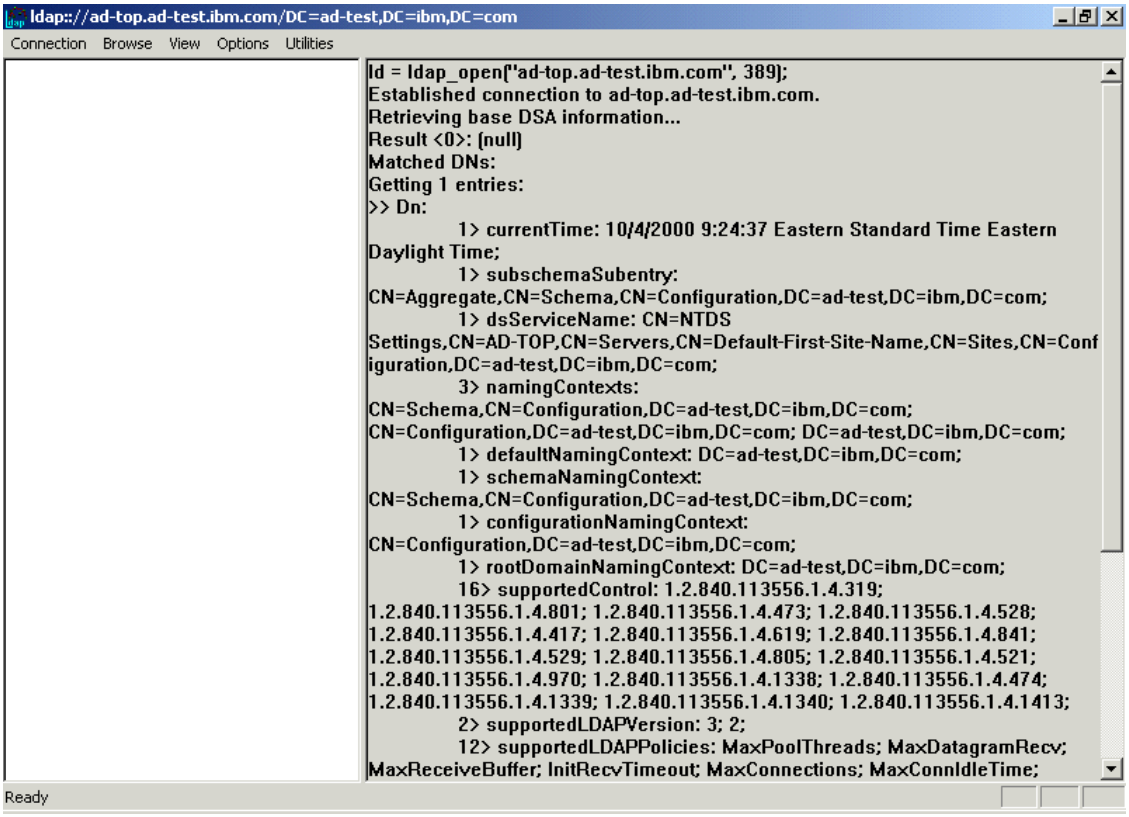

Figure 37. The Active Directory RootDSE

6. The next step is to authenticate, or *bind*, yourself to the directory server. To do this, select **Connection** and then **Bind**, as shown in Figure 38.

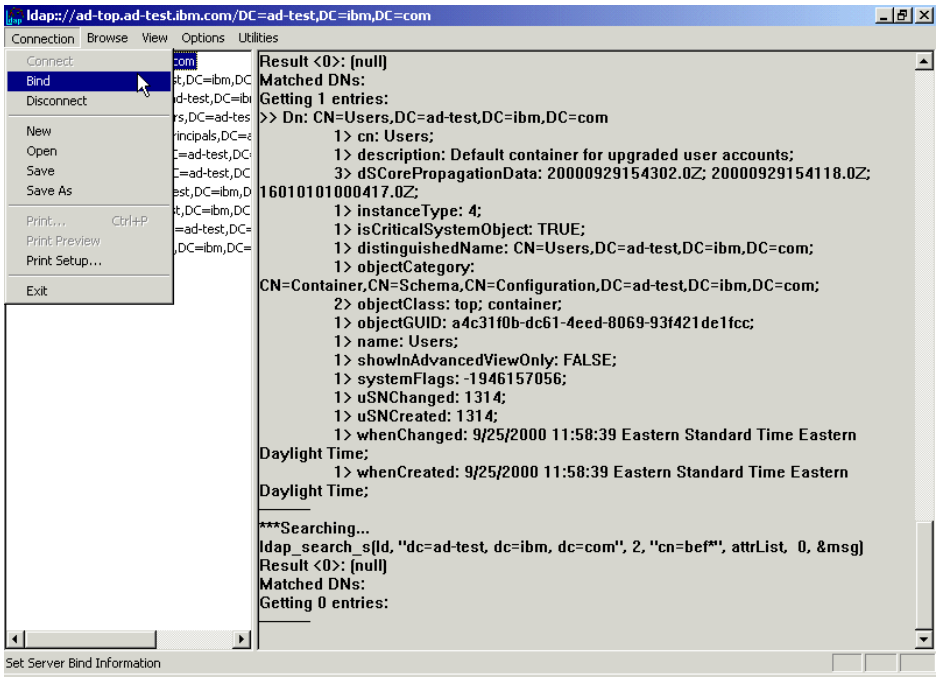

Figure 38. Binding to the directory server

7. The Bind dialog box will be displayed, as shown in Figure 39. Supply credentials that have enough permissions to perform the intended operations. You may also choose to click the **Advanced** button for other options, such as simple vs. extended binds.

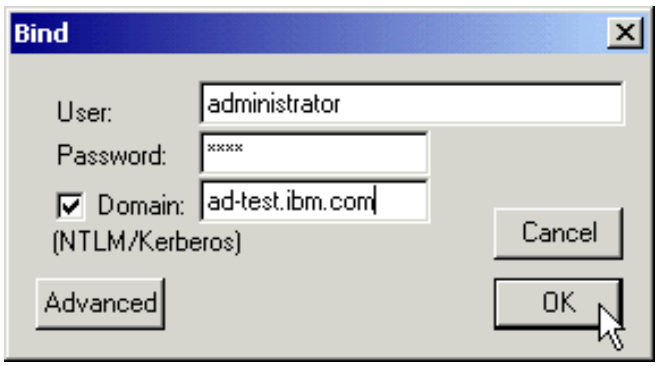

Figure 39. The Bind dialog box

8. The final step you may want to perform before actually executing the search is to display a tree view on the left-hand side of the window. This is actually an optional step, but it simplifies the query process and it also reduces typing (and the chance of making typing mistakes). To display a tree view, select **View**, and then **Tree**, as shown in Figure 40.

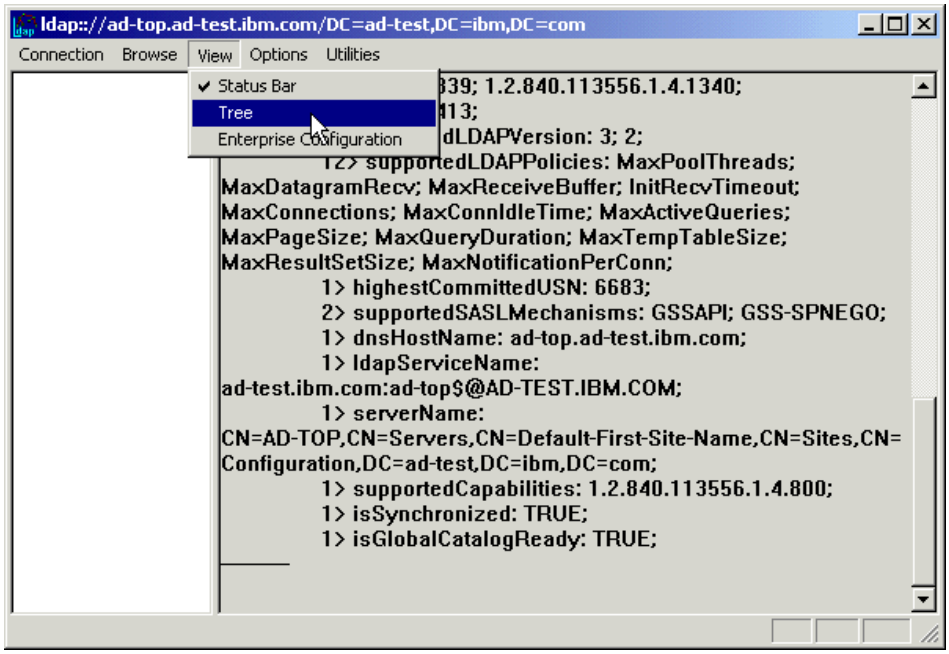

Figure 40. Setting up the tree view

9. In the Tree View dialog box (Figure 41), you should type the common name of the naming context you want to display in the tree view. In this case, we wanted the default user naming context.

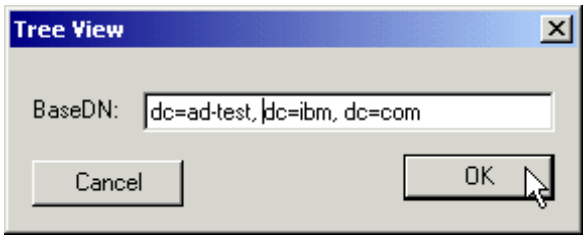

Figure 41. Choosing the naming context to be displayed in the tree view

10. The tree view will be displayed on the right-hand side of the window, as shown in Figure 42.

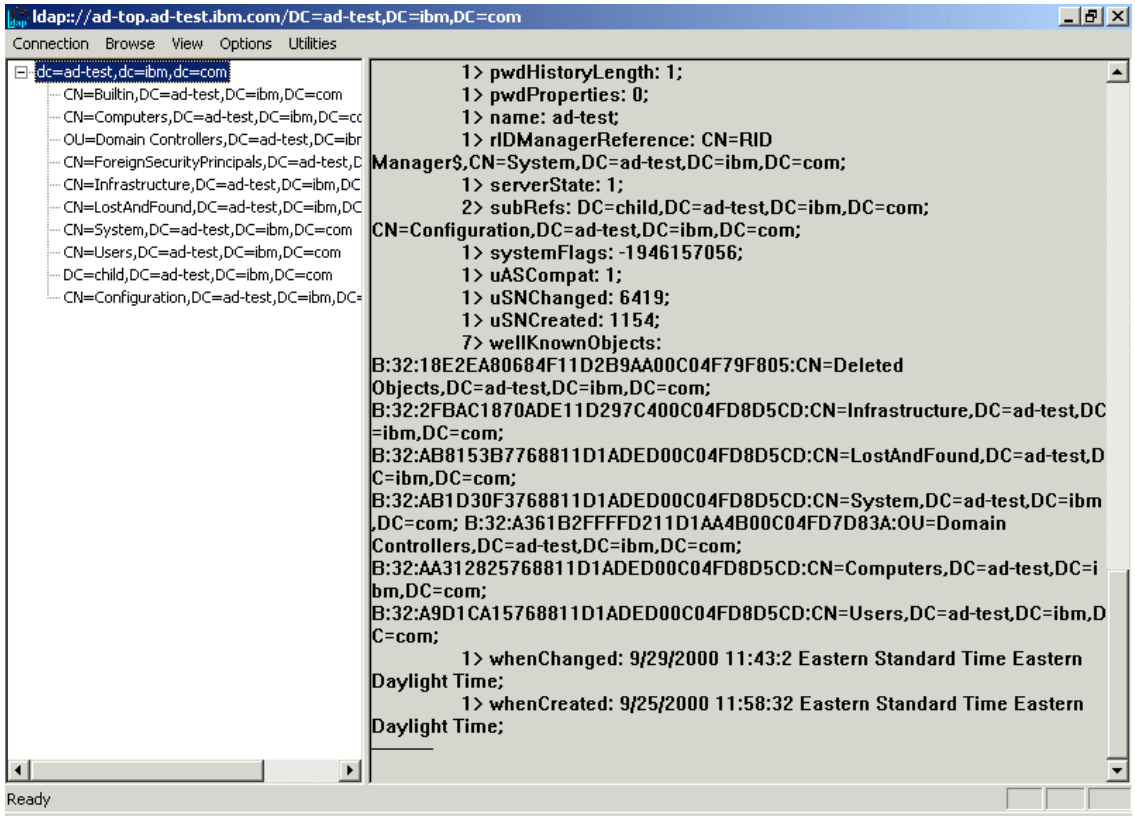

Figure 42. The tree view

11. Finally, we are ready to perform the search. To accomplish this, we can either right-click the tree view and select **Search**, or select **Browse**, and then **Search**, as shown in Figure 43.

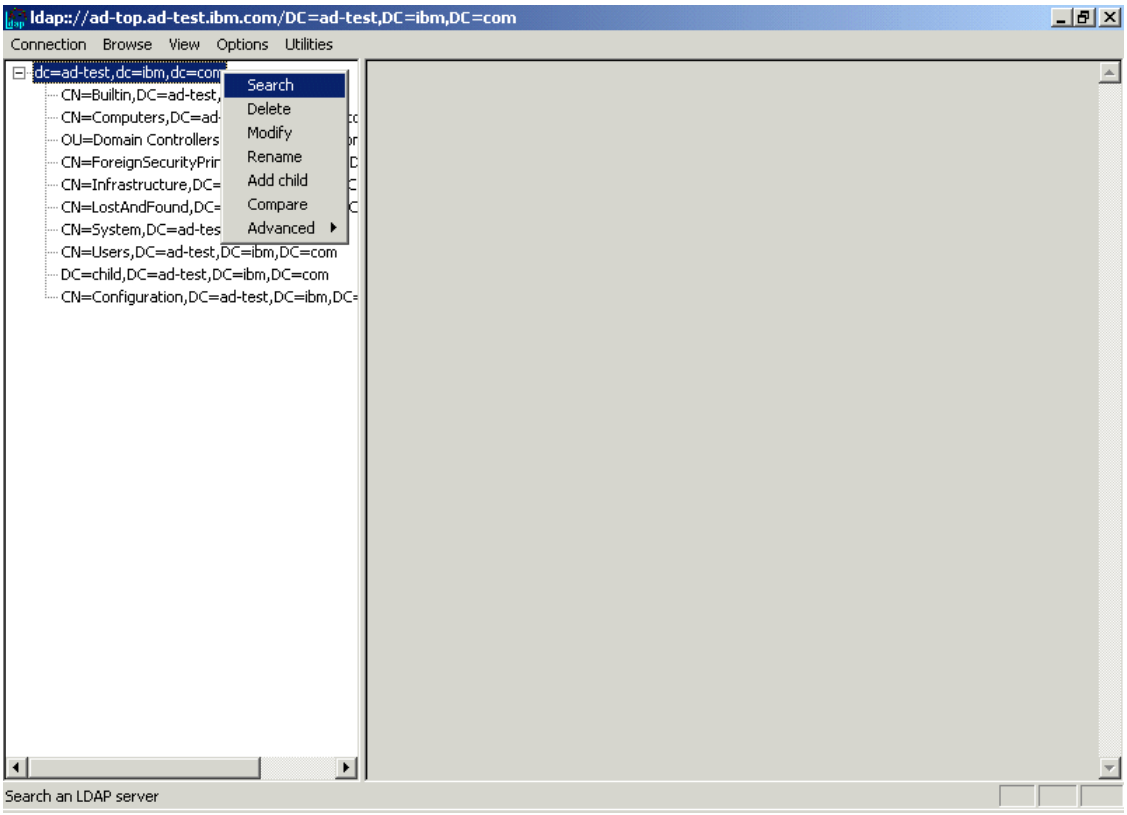

Figure 43. Initiating a search

12. In the Search dialog box that follows (Figure 44), you should type the base DN (the starting point) for this search and the criteria for this search, such as `objecttype=*` or `cn=admin*`. You should also select the scope: **base** (just the base DN), **one level** (the base DN plus its immediate children) or **subtree** (everything under the base DN, no matter how many levels deep).

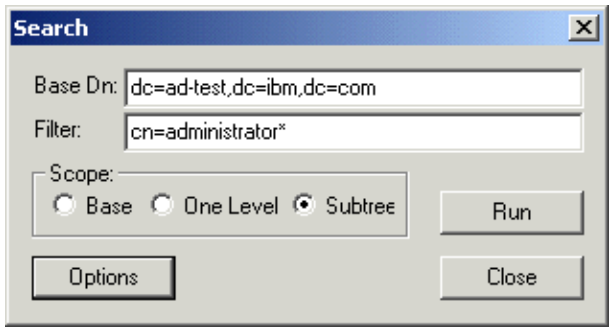

Figure 44. The Search dialog box

If you have a very large directory, you may want to click the **Options** button and set time and size limits, choose a different search call type, or disable the **Chase referrals** option in the Search Options dialog (Figure 45).

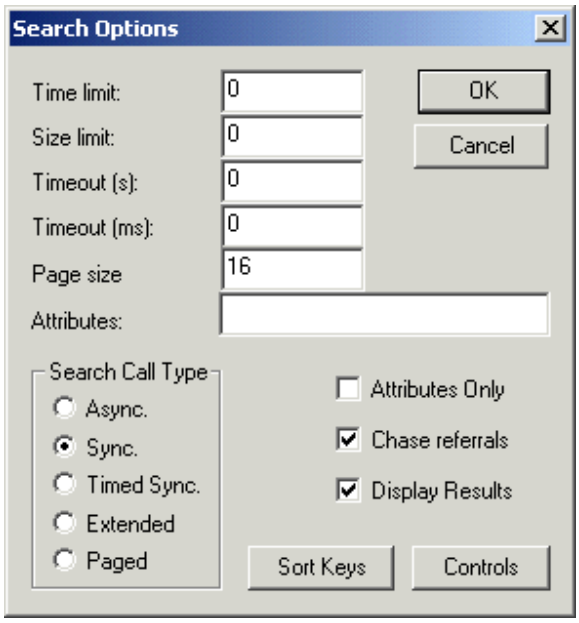

Figure 45. The Search Options dialog box

13.You can see the results of the search in Figure 46:

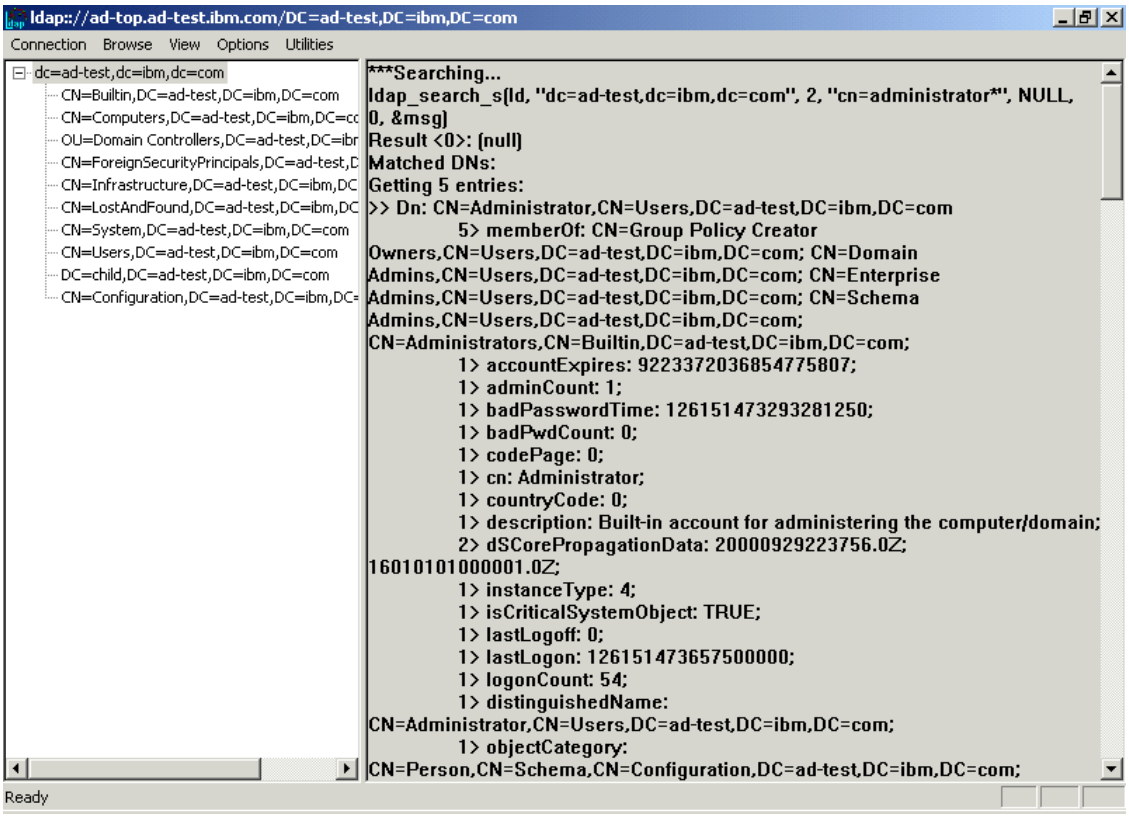

Figure 46. Search results

2.6.2 Searching for SecureWay data through Active Directory

The next step was to test the referrals to the SecureWay Directory. This means that the client would connect to the Active Directory server and query for data stored in the SecureWay Directory.

To evaluate and test this case, we executed all steps from 2.6.1, “Searching for data in the Active Directory” on page 61, but instead of specifying a base DN that belongs to the naming context stored in the Active Directory server, we specified a base DN belonging to the naming context stored in the SecureWay Directory, as shown in Figure 47.

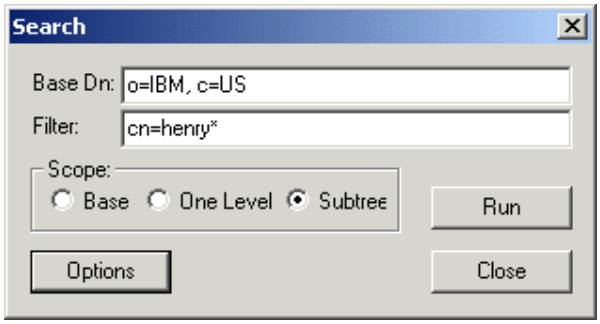

Figure 47. Searching for SecureWay data from Active Directory

The results you receive will depend on whether the **Chase Referrals** option is enabled in the Search Options dialog box. In our sample search, the result received when the option was disabled was the actual referral:

```
***Searching...
ldap_search_s(ld, "o=IBM,c=US", 2, "cn=henry*", NULL,  0, &msg)
Error: Search: Referral. <10>
Result <10>: 0000202B: RefErr: DSID-031005EE, data 0, 1 access points
    ref 1: 'ldap1.ad-test.ibm.com'

atched DNs:
Getting 0 entries:
```

When this option is enabled, on the other hand, the client application will chase the referral and fetch the data correctly:

```
***Searching...
ldap_search_s(ld, "o=IBM,c=US", 2, "cn=henry*", NULL,  0, &msg)
Result <0>: (null)
Matched DNs:
Getting 1 entries:
>> Dn: cn=Henry Nguyen, ou=Austin, o=IBM, c=US
    3> objectclass: person; organizationalPerson; top;
    1> cn: Henry Nguyen;
    1> sn: Nguyen;
    1> telephonenumber: 1-812-855-4028;
    1> internationalisdnnumber: 755-4028;
    2> facsimiletelephonenumber: 1-812-855-9087; 755-9087;
    1> title: AIX Support;
    1> postalcode: 9551;
```

Note

Throughout this section, we have used disjointed namespaces. It is also possible to use a contiguous namespace in both directories, and use the same base DN for all queries.

2.6.3 Searching for data in the SecureWay Directory

The simplest SecureWay Directory case occurs when the client connects to a server and queries for data stored in the same server.

To evaluate and test this case, we executed all steps from 2.6.1, “Searching for data in the Active Directory” on page 61, with the following differences:

- In step 4 on page 63, we have supplied the address to the SecureWay Directory server. This is shown in Figure 48.

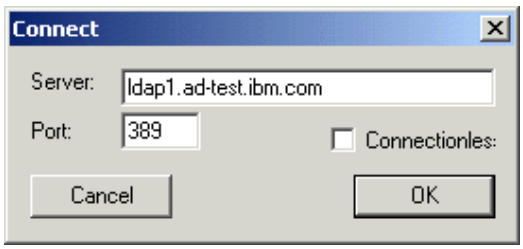

Figure 48. Connecting to a SecureWay server

This produced the following RootDSE:

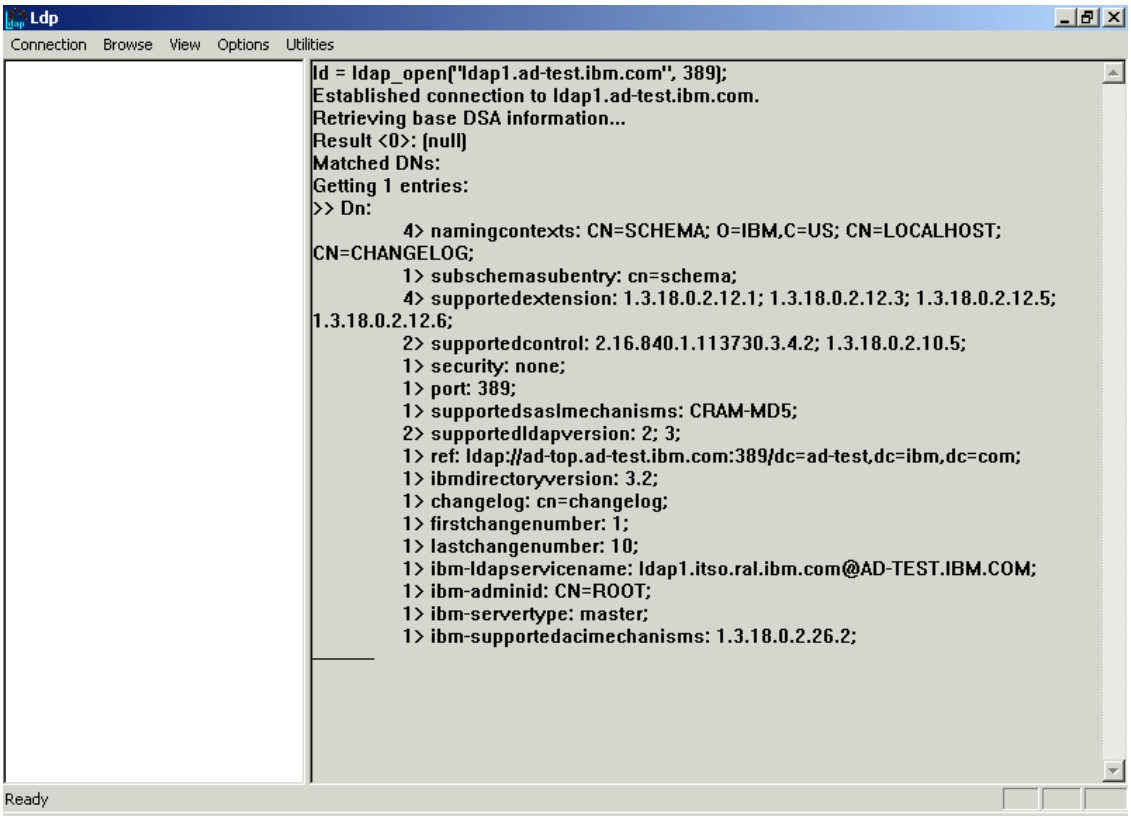

Figure 49. The SecureWay RootDSE

- In step 7 on page 65, we have supplied the credentials to bind to the SecureWay Directory server, as shown in Figure 50.

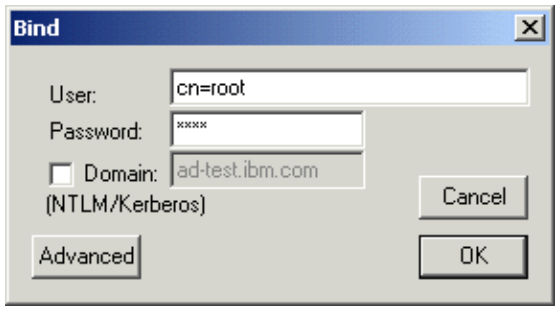

Figure 50. Binding to SecureWay

- In step 9 on page 66, we have supplied the naming context stored in the SecureWay Directory server. This is shown in Figure 51.

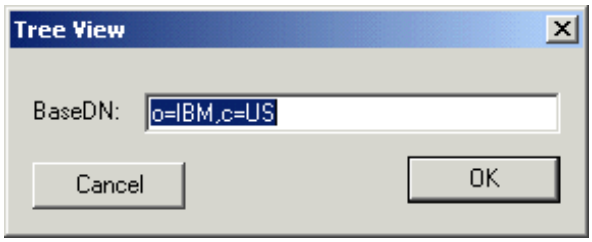

Figure 51. The SecureWay naming context

These actions produced the results displayed in Figure 52.

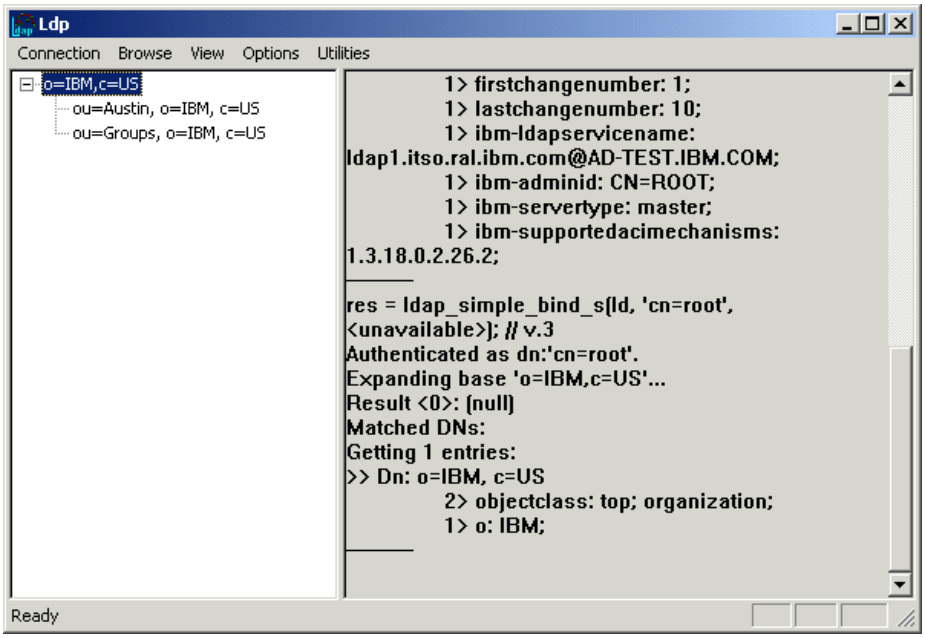

Figure 52. The SecureWay tree view

- Finally, in step 12 on page 68, we have supplied the base DN of the SecureWay Directory server naming context in the Search dialog box (Figure 53).

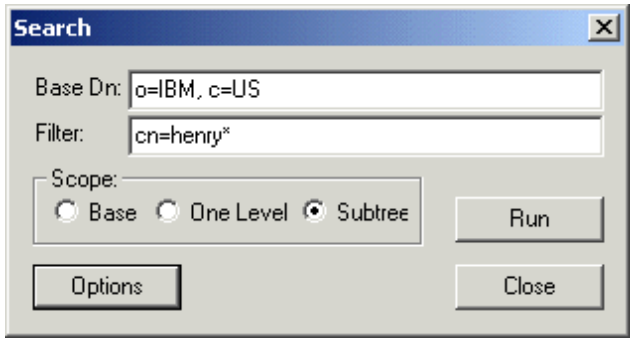

Figure 53. Searching the SecureWay Directory

The results of this search can be seen in Figure 54.

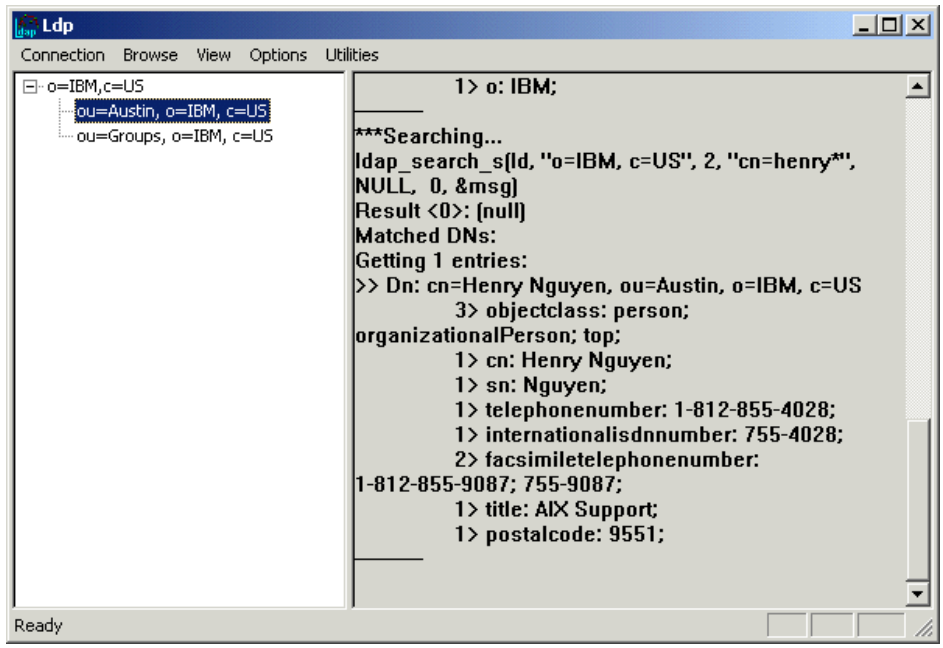

Figure 54. The SecureWay search results

2.6.4 Searching for Active Directory data through SecureWay

The last case we wanted to test was referrals to the Active Directory. This means that the client would connect to a SecureWay Directory server and query for data stored in the Active Directory.

To evaluate and test this case, we executed all steps from 2.6.3, “Searching for data in the SecureWay Directory” on page 72, but instead of specifying a base DN that belongs to the naming context stored in the SecureWay server, we specified a base DN in the Active Directory naming context.

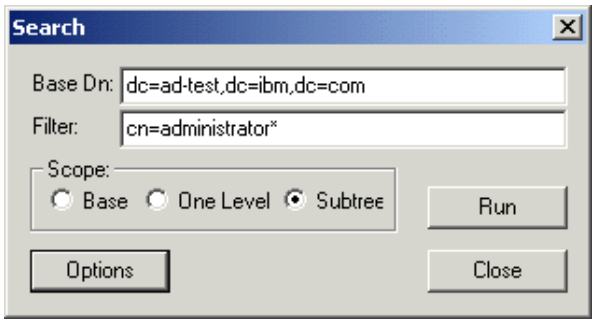

Figure 55. Searching for Active Directory data from SecureWay

The results you will receive will depend on whether the **Chase Referrals** option is enabled in the Search Options dialog box. In our sample search, the result received when the option was disabled was the actual referral:

```
***Searching...
ldap_search_s(ld, "dc=ad-test,dc=ibm,dc=com", 2, "cn=admin*", NULL,  0, &msg)
Error: Search: Referral. <10>
Result <10>:
Matched DNs:
Getting 0 entries:
```

When this option is enabled, on the other hand, the client application will chase the referral and fetch the data correctly:

```
***Searching...
ldap_search_s(ld, "dc=ad-test,dc=ibm,dc=com", 2, "cn=admin*", NULL,  0, &msg)
Result <0>: (null)
Matched DNs:
Getting 1 entries:
>> Dn: CN=Administrators,CN=Builtin,DC=ad-test,DC=ibm,DC=com
    5> member: CN=Domain Admins,CN=Users,DC=child,DC=ad-test,DC=ibm,DC=com;
CN=Domain Admins,CN=Users,DC=ad-test,DC=ibm,DC=com; CN=Enterprise
Admins,CN=Users,DC=ad-test,DC=ibm,DC=com;
CN=BeF,CN=Users,DC=ad-test,DC=ibm,DC=com;
CN=Administrator,CN=Users,DC=ad-test,DC=ibm,DC=com;
    1> cn: Administrators;
    1> description: Administrators have complete and unrestricted access to the
computer/domain;
    2> dSCorePropagationData: 20000929154302.0Z; 16010101000001.0Z;
    1> groupType: -2147483643;
    1> instanceType: 4;
    1> isCriticalSystemObject: TRUE;
    1> distinguishedName: CN=Administrators,CN=Builtin,DC=ad-test,DC=ibm,DC=com;
    1> objectCategory:
CN=Group,CN=Schema,CN=Configuration,DC=ad-test,DC=ibm,DC=com;
    2> objectClass: top; group;
    1> objectGUID: 93e0a946-928a-4168-aee6-0ec1fafe4348;
    1> objectSid: S-20-220;
    1> name: Administrators;
    1> sAMAccountName: Administrators;
    1> sAMAccountType: 536870912;
    1> showInAdvancedViewOnly: FALSE;
    1> systemFlags: -1946157056;
    1> uSNChanged: 3818;
    1> uSNCreated: 1418;
    1> whenChanged: 9/26/2000 16:5:17 Eastern Standard Time Eastern Daylight Time;
1> whenCreated: 9/25/2000 11:58:41 Eastern Standard Time Eastern Daylight Time;
```

Note

Throughout this section, we have used disjointed namespaces. It is also possible to use a contiguous namespace in both directories, and use the same base DN for all queries.

2.7 What we have discovered

Once SecureWay Directory and Active Directory were properly configured and set up to do referrals, we were able to search for data in either directory no matter which server we connected to. Therefore, referrals make it possible to have SecureWay Directory and Active Directory co-exist in the same environment.

In our fictional scenario, this means the new company (the result of the merger) could use this functionality to enable a limited coexistence for the company's applications, while both directories undergo changes resulting from the merger process.

Chapter 3. Scenario2: Adding Domino

The following scenario is of a fictional company that has defined the IBM SecureWay Directory as its enterprise directory. This company also has a Windows 2000 domain, and it uses Lotus Domino for e-mail. In this scenario we explore referral functionality by having an LDAP client searching for data stored on a directory (the Domino directory) by connecting to another directory (Active Directory). SecureWay will act as the hub directory, so all directories will refer to the SecureWay server.

Note

The Lotus Domino R5 product includes an LDAP Version 3-compliant Directory.

3.1 About our test environment

For this scenario, we have set up a computing environment very similar to the one in Chapter 2, “Scenario1: Integrating SecureWay with Active Directory” on page 45, consisting of one Windows NT 4.0 server running the SecureWays product and one Windows 2000 domain controller, plus an additional Windows 2000 server running Lotus Domino R5 as shown in Table 3:

Table 3. Test environment configuration

Machine Name	LDAP1	AD-TOP	AD-BOT
Operating system	Windows NT Server 4.0 SP6a	Windows 2000 Advanced Server SP1	Windows 2000 Advanced Server SP1
Domain	(none)	ad-test.ibm.com	ad-test.local
Domain role	N/A	domain controller	domain controller
Additional software	IBM SecureWay Directory V3.2	(none)	Lotus Domino R5
	DB2 UDB V6.1.0.21		
	Java Developer Kit (JDK) V1.1.8		

Machine Name	LDAP1	AD-TOP	AD-BOT
Additional Windows components	Microsoft IIS V3.0	Microsoft IIS V5.0	(none)
		Active Directory	

Since the IBM SecureWay Directory server performs the role of enterprise directory in this scenario, we have adopted a hub-and-spoke architecture, with server LDAP1 being the hub. The scenario environment is illustrated in Figure 56:

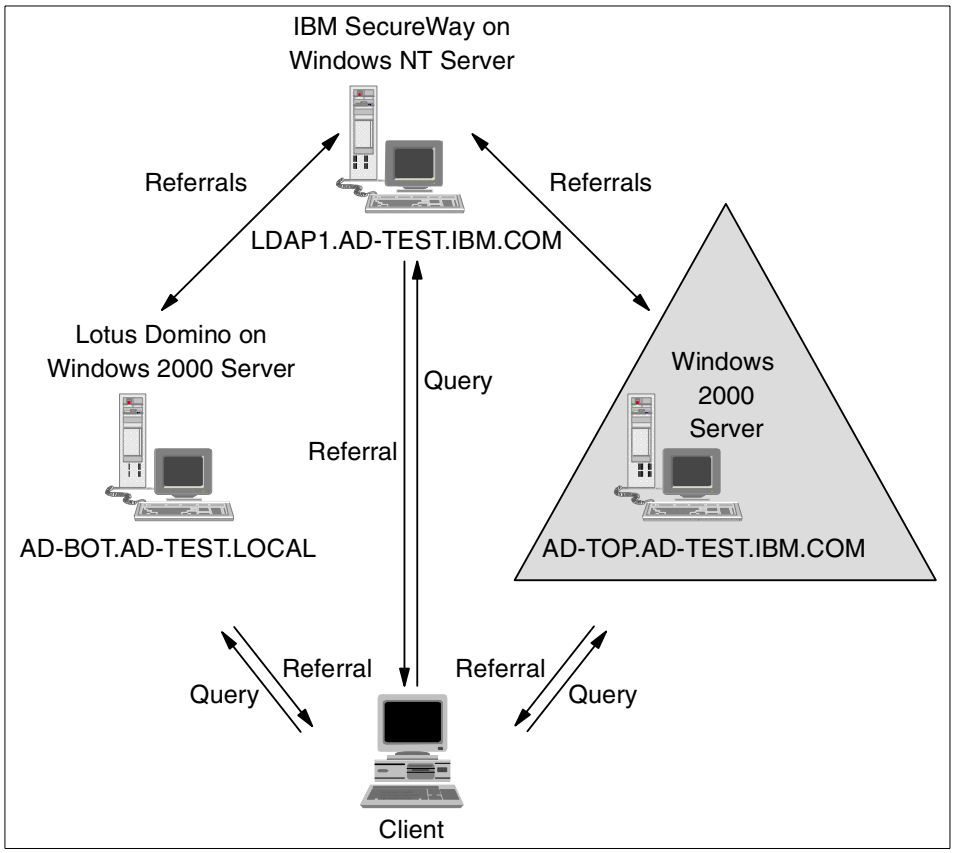

Figure 56. Adding Domino to the enterprise directory

For the sake of simplicity, AD-TOP is the primary DNS server for all forward and reverse DNS zones, and all zones are standard zones.

3.2 How we tested

For this scenario, we decided to build upon the previous scenario by extending its scope to include the Lotus Domino directory, but instead of creating everyone-to-everyone referrals, we decided to create a referral structure more similar to the one we expect to find in a corporate environment: an enterprise directory (which may or may not be distributed and/or replicated among several sites) and the NOS- and application-specific directories, which have referrals to the enterprise directory only.

3.3 Configuring SecureWay

For instructions on installation and configuration of SecureWay refer to the *LDAP Implementation Cookbook*, SG24-5110. Once all the software is installed Secureway needs to be customized. The steps in 2.3, “Configuring SecureWay” on page 46 outline what was done.

This time, we also created a referral to the Domino directory:

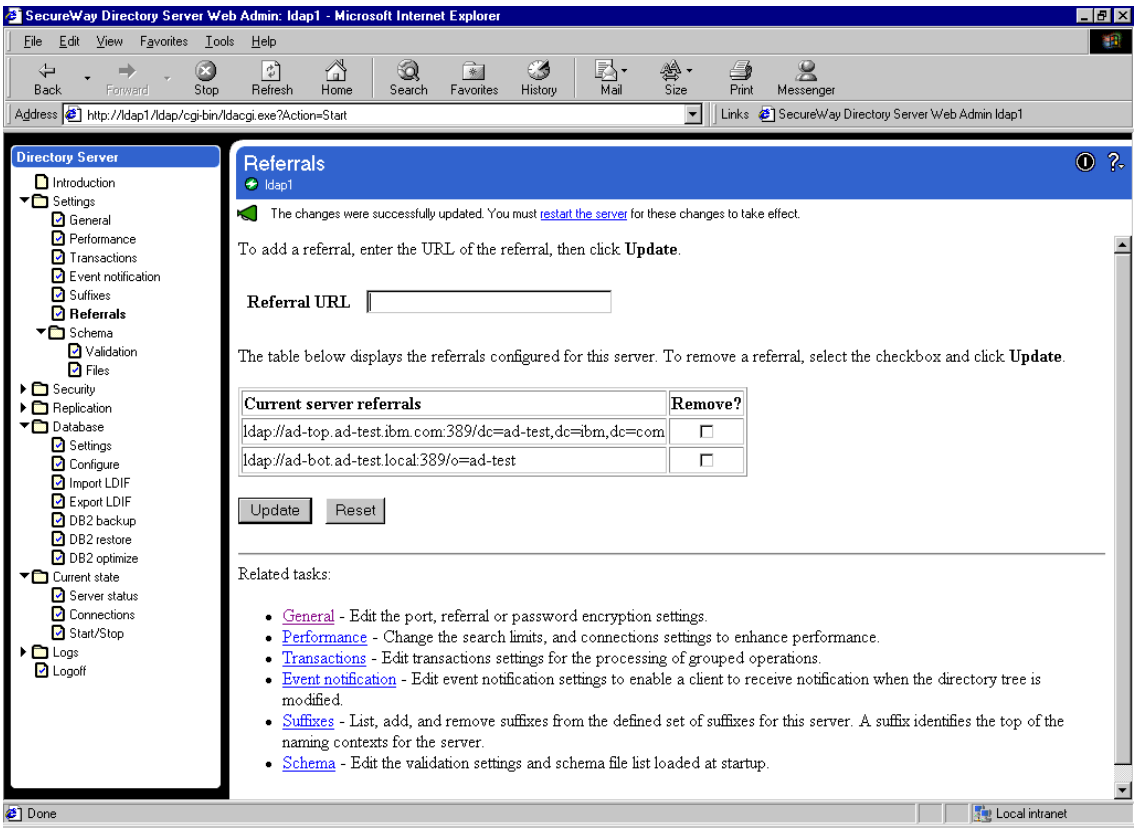

Figure 57. Creating a referral to the Domino directory in SecureWay

Note

For referrals to work you need to ensure that there is sufficient permission to access the directory tree. Directory tree security may be updated through the IBM SecureWay Directory Management Tool.

3.4 Creating a referral in Active Directory

To configure the Active Directory to do referrals to the external naming contexts, we have used the ADSI Edit MMC snap-in. The steps in 2.5, “Creating a referral in Active Directory” on page 50 detail this process.

An important note on security

For referrals to work, the security on all directories should be configured in such a way that permissions are enough for all directory services performing the desired operations on each other. Such configuration goes beyond the scope of this book.

3.5 Configuring Lotus Domino for referrals

Lotus Domino's approach on LDAP is somewhat different from the two directory services reviewed previously in this book. Domino implements LDAP as a means for other directories to access its directory information, as a means of authenticating Web clients, and as a means for Lotus Notes clients to validate e-mail or authentication information against external directories.

3.5.1 Setting up the LDAP service on a Domino server

If you are adding LDAP services to an existing Domino server, all you need to do to get the service up and running is type `load LDAP` from a server command prompt. In addition, you will want to modify the "ServerTasks=" line in notes.ini by adding "LDAP" to the list of tasks. This will ensure that the LDAP service is automatically loaded at server startup.

Note

For more information on setting up Domino directories, please see *Getting the Most From Your Domino Directory,* SG24-5986.

3.5.2 Configuring the LDAP service

To make configuration changes to your LDAP server, you will need to create a server configuration document. From the Administrator client, choose the **Configuration** tab, expand the **Server** section, and click the **Configurations** icon. Now click the **Add Configuration** action button to create a new configuration document.

Configuration documents that are used by the LDAP server must have the **Use these settings as the default settings for all servers** checkbox selected. Note that there can be only one configuration document that is thus designated. Once you have selected this option, the LDAP tab appears. Click it to move to that section, and you should see a window like Figure 58.

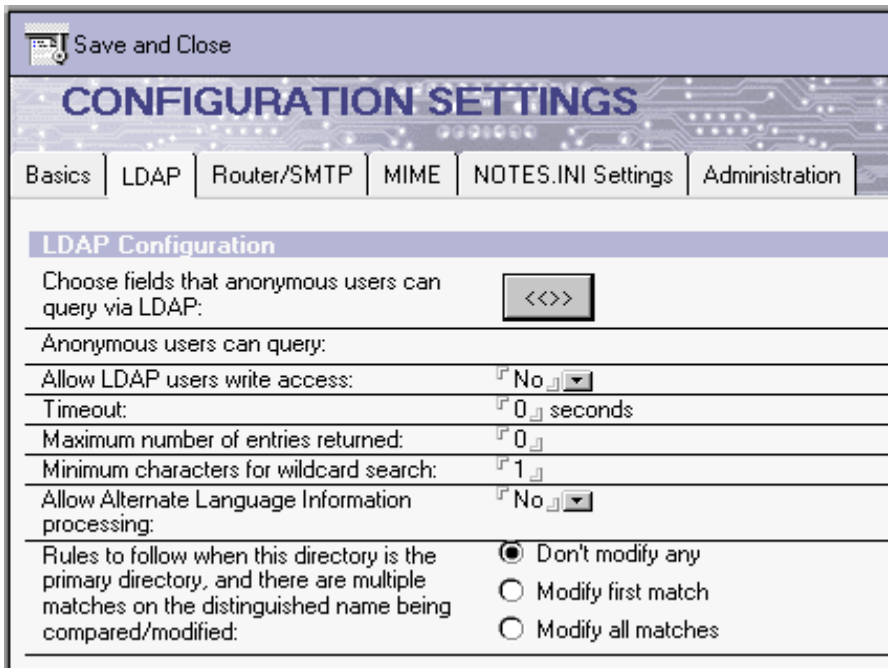

Figure 58. LDAP configuration settings

Clicking the **<<>>** button will bring up a dialog in which you can add fields to the default fields that anonymous users can query. You can choose from the available forms in the Domino directory and add fields to those exposed to anonymous users by clicking the **Show Fields** button and selecting fields to add. You can also remove fields from the list of those available to anonymous users through this dialog. See Figure 59.

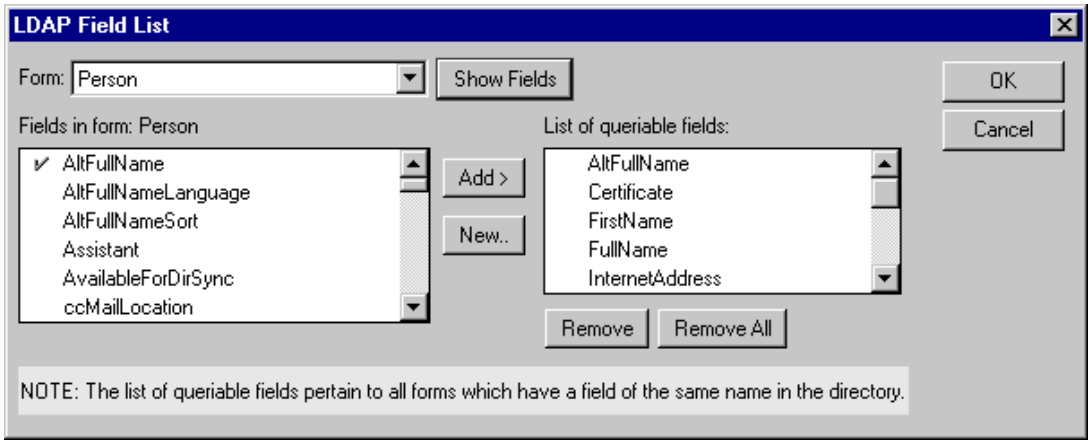

Figure 59. Modifying fields available to anonymous users

The Allow LDAP users write access field is a global setting that will prevent any LDAP Modify, Add, Delete, and Modify RDN operations from changing the Domino directory. Note that the LDAP service takes full advantage of Domino's security model. Adding new directory entries requires Editor access in the ACL; modifying existing entries requires Editor or Author (with appropriate modification rights); deleting existing entries requires the Delete Documents ability in the ACL. Of course, this setting does not affect anyone's ability to search the LDAP directory.

The next three settings are for performance tuning the LDAP service.

- **Timeout** specifies the number of seconds to wait before automatically terminating a search request. The default, "0" allows for an unlimited time.
- **Maximum number of entries returned** specifies how many results can be returned by a search operation. Again, "0", the default, specifies no limit.
- **Minimum characters for wildcard search** allows the administrator to require the user to provide more specifics, thus reducing the result set from a Search operation. The default is "1".

The next setting determines whether the LDAP service will serve up alternate language information. This information will be UTF8-encoded Unicode. Note that this works only if you have defined alternate language information in the directory. See *Administering a Domino Server,* part numbers CT7VHNA and CT7VINA, found at `http://www.lotus.com` on how to set up alternate language information for end users.

The last setting governs how the LDAP service handles requests when the DN operated upon is not unique. This applies to modify, delete, and compare operations, or when an add operation is attempted when there is already an entry in the directory. Selecting **Don't modify any** will prevent the operation from concluding. Choosing **Modify first match** will modify, delete, or compare with the first match found. Selecting **Modify all matches** will perform the operation on all matches found. Note that if your server is using Directory Assistance to unify searches for multiple Domino directories, this setting will apply to all directories handled by this server, if the user has the required permissions to those directories.

Note

When setting up referrals in Lotus Domino, keep in mind that Domino will always search secondary directories first, before searching LDAP directories. Also, make sure you have set up the LDAPReferrals entry on the notes.ini file. Otherwise, your searches will receive only one referral. For more information on this, please refer to the Lotus Domino R5 product documentation.

3.5.3 Setting up and configuring Directory Assistance

In Domino terminology, a referral to another directory is called Directory Assistance. The first step in setting up Directory Assistance is to create the Directory Assistance database. From a Notes client, choose **File ->Database -> New...** Click the **Template Server** button and choose an appropriate server that has the Directory Assistance database template on it. Type an appropriate title for the database, such as Acme Corp.Directory Assistance and a simple filename, such as AcmeDA.nsf. From the list of templates choose **Directory Assistance** and click **OK** (see Figure 60). Domino will create the database with your name in the ACL as Manager.

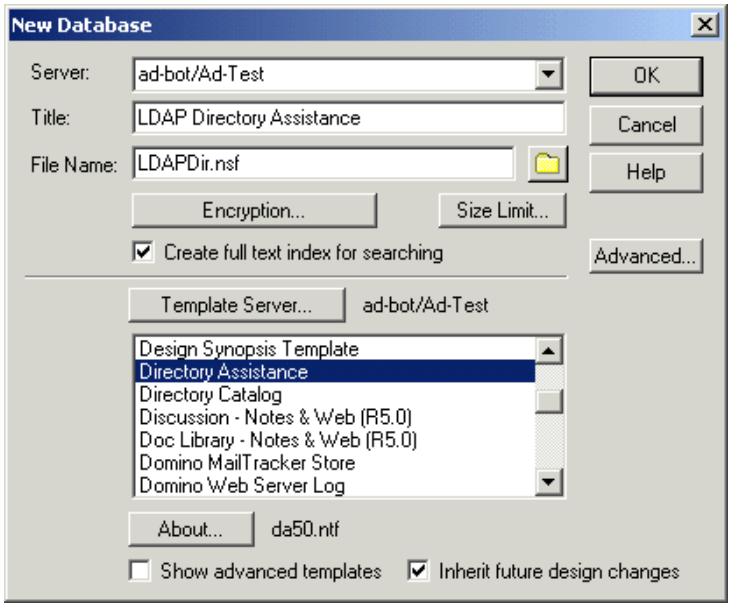

Figure 60. Creating the Directory Assistance database

The next step is to create rules in the Directory Assistance database to help servers search the proper directories to find entries. The newly created Directory Assistance database should be open in Notes. From the Directory Assistance view, click the **Add Directory Assistance** action button. This will bring up a new Directory Assistance document.

This Directory Assistance database can be used by Notes clients to look up mail addresses, by Web browsers to authenticate to a Domino server using an external directory, or by LDAP clients and/or applications looking for LDAP information, which will receive a referral to the new directory.

The Basics tab of the Directory Assistance document has information to help administrators organize the rules in the Directory Assistance database. For an LDAP rule, the domain type will be LDAP, as you can see in Figure 61.

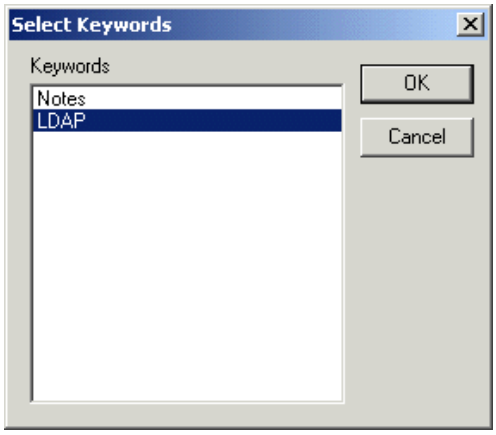

Figure 61. Choosing the domain type

A domain name is a unique identifier that you can assign to each directory. It doesn't have to be the same name as the actual Notes domain described by the directory, but you might want to use that as a guide. If you enter a domain name that is not unique in the Directory Assistance database, you may get the error `User not found in any Name and Address Book` when trying to search the secondary directory. The Company field can be used to keep track of domains whose names don't obviously point to one company. The Search Order field will determine the order in which each directory is searched for matches. The Enabled setting provides a mechanism for quickly enabling and disabling lookups to foreign directories. The Group Expansion field allows you to specify one LDAP directory that Domino will use for groups. This setting can only be used for one LDAP directory. You can see the final Basics tab in Figure 62:

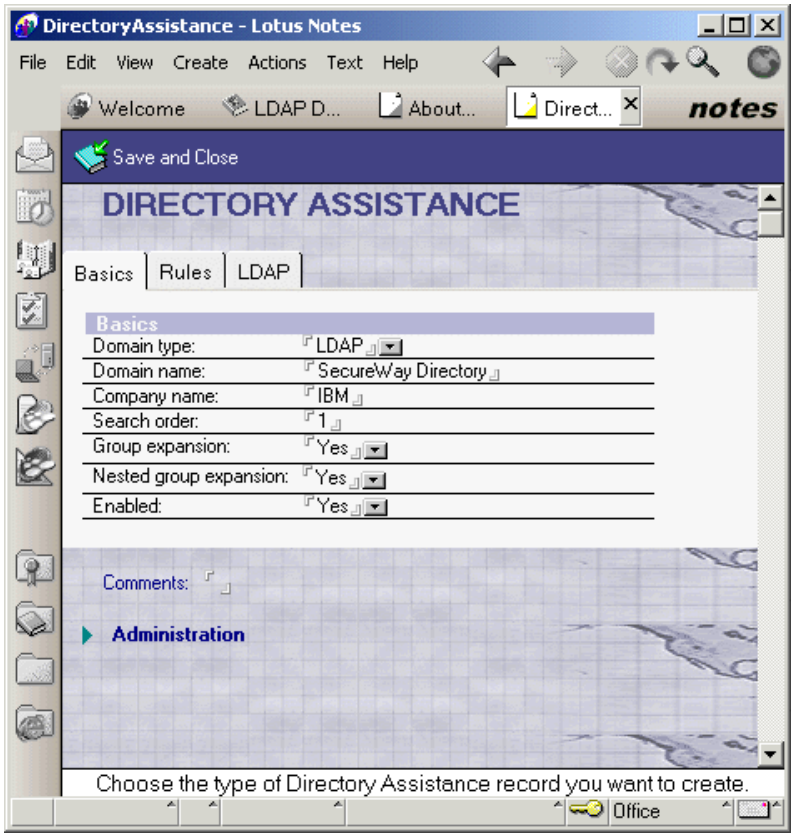

Figure 62. Directory assistance's Basic tab

The rules that define when Directory Assistance should consult another directory are specified on the Rules tab. Once you've finished filling out the Basics tab, click the **Rules** tab and specify what context applies to this directory. If you want this directory to be searched regardless of the name given for a match, you can put in a rule with all asterisks. More likely, a given directory will hold information for people in given organizations, so you can fill in the organization field. If you want to include a specific OU, you can list it in the relevant OU field(s), you can use a wildcard (*) character to include all OUs, or you can leave the field blank to prevent matching entries with an OU in that position. Note that LDAP rules can only be used to authenticate Web users to a Domino Web server, not Notes users.

After you define your matching rules, you can specify whether that rule is enabled (again, to provide an easy mechanism to quickly enable/disable matching), and whether or not it can be used for authentication to the server

("trusted for credentials"). A rule that is not trusted can still be used for mail addressing from a Notes client, and so forth, but will not govern authentication.

The LDAP tab of the Directory Assistance (Figure 63) document is where you can specify the LDAP-specific parameters for this rule. You first need to specify the host name of the server that is running the LDAP server. In addition, if the external server requires a user name and password to bind to it, enter those in the next two fields. We recommend that you encrypt the fields containing the user name/password combination so that they won't be stored in the clear. See the following note for more information on how to do this. Many LDAP servers require you to provide a base DN (such as "o=IBM,c=US") to search upon. If your LDAP server requires this, enter the base DN in the field provided.

Encrypting fields

To encrypt the Directory Assistance document and thereby restrict access to it, choose **File ->Document Properties**. Click the **Security** tab (the one with the little key icon) and choose the people who should have access to the document by clicking the person icon next to the Public Encryption keys.

The next two check boxes describe what clients you want to be allowed to use this directory. Notes clients (for mail lookup) and Web clients (for authentication to Domino servers) are selected by default. If you want to allow LDAP clients to use the external directory (by way of a referral) you should check that box as well.

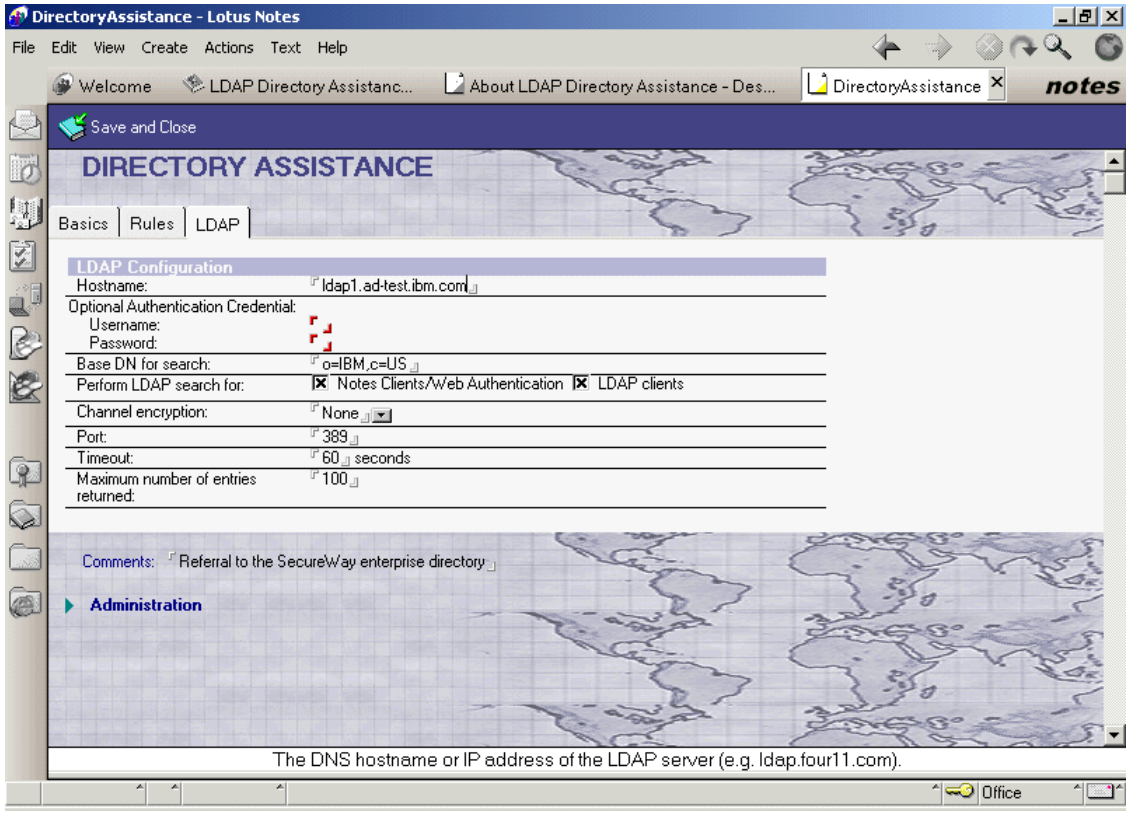

Figure 63. Directory Assistance's LDAP tab

In order to secure the channel between the Domino server and the external LDAP server, it is recommended that you enable SSL in the Channel encryption field. This will encrypt the packets sent across the wire between the two servers and prevent anyone from capturing directory data. If you want your Domino server not to connect when the server on the other end has an expired certificate, select that option. Unless you have unusual difficulty getting the Domino server to negotiate the SSL protocol with the external LDAP server, you should probably choose the negotiated setting for the SSL protocol version. This will allow the two servers to freely negotiate what SSL level to use to secure the channel. Finally, you can add one more layer of security by requiring that the SSL certificate presented by the server match its domain name by selecting **Enabled** in the Verify server's name with remote server certificate field. This field will only appear when the channel encryption is enabled and thus is not shown in Figure 63.

The last two fields on this form are for performance-tuning purposes. Here you can set the timeout for queries to the external directory (the default is 60 seconds). You can also set a maximum number of entries returned (default is 100). If you are having trouble getting a timely response from the external server, you may want to adjust these settings.

3.5.4 Deploying Directory Assistance

Once you have set up your matching rules, you will need to make the Domino server aware of the Directory Assistance database. To set that up, you will need to add its file name to the server's Server Document in the Domino directory. To do this, from the Administrator client, choose the **Configuration** tab, then expand the **Server** section in the left-hand pane and click the **Current Server Document** icon. On the **Basics** tab (Figure 64), you will see a place to enter the file name of the Directory Assistance database. Enter the name, save the document, and then reboot the server.

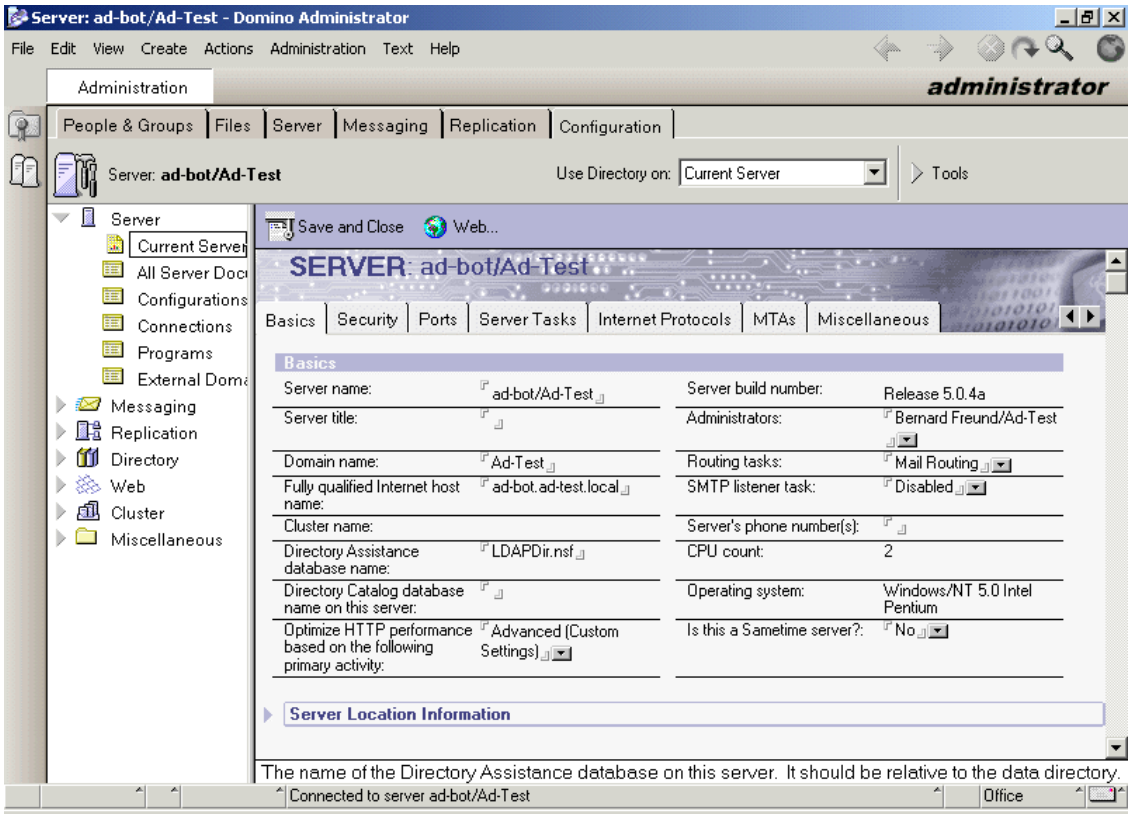

Figure 64. Deploying Directory Assistance

3.5.5 Setting up directory catalogs

Before you set up a directory catalog, make sure that your directory profile is set up correctly. To do this, open the directory and choose **Actions -> Edit Directory Profile**. Make sure that there is an entry in the Domain defined by this public directory field, and that it is unique among directories that will be aggregated. If this directory is used for Notes mail users, it should be the same as the mail domain for those users. You will need to take this step for every directory that is going to be aggregated. If you have directories that are using an older version of Domino, you should upgrade the design of those databases to the Domino R5 directory template. If you do not have a directory profile, you will encounter difficulties if groups in more than one directory have the same name. Domino will not be able to distinguish between the two and your mail might get routed to the wrong set of people.

Additionally, you will need to make sure that you have replicas of every directory to be aggregated on the server. While it is possible to have the directory catalog gather entries from remote servers, it is not recommended. Creating the catalog will take significantly longer. Remember to create connection documents back to the servers you replicate the other directories from, to ensure that the data is kept up-to-date.

Now you need to create the database that will hold the aggregated directories. From a Notes client, choose **File -> Database -> New**... Choose the server you will be using to aggregate the directories. Type in a descriptive name for the directory catalog, such as LDAP directory catalog. Type in a concise file name, such as ldapdircat.nsf (Figure 65).

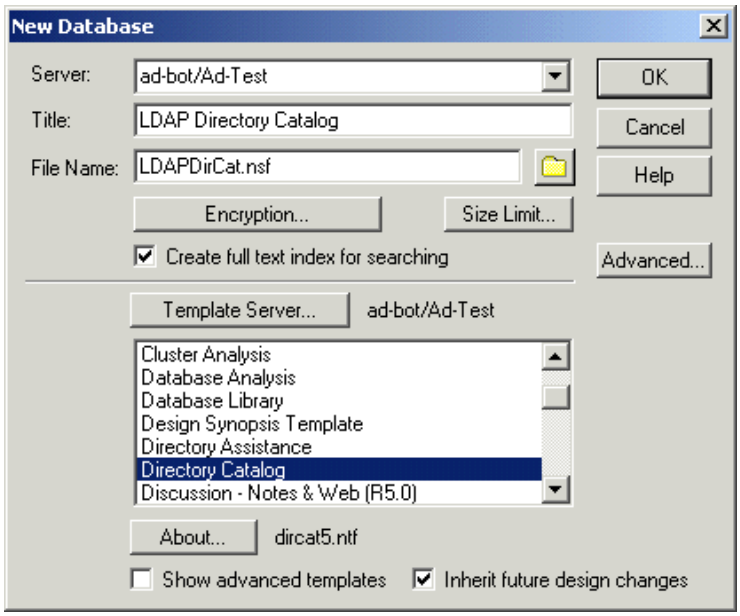

Figure 65. Creating the directory catalog database

Click the **Template Server...** button and choose a server with the **DirCat** template, select the **Directory Catalog** template and click **OK**. Select **Yes** when prompted to create a full-text index for the directory catalog. A full-text index is especially needed if this server will be handling LDAP queries.

Note

Be careful not to select the Directory Assistance or Catalog (5.0) templates instead of directory catalog as shown in Figure 65.

3.5.6 Configuring the directory catalog

Now that you've created the database, open it and choose **Create -> Configuration**. In this form, specify what directories you want to aggregate into this directory catalog, and some additional configuration parameters. The form you see should look like Figure 66 on page 95.

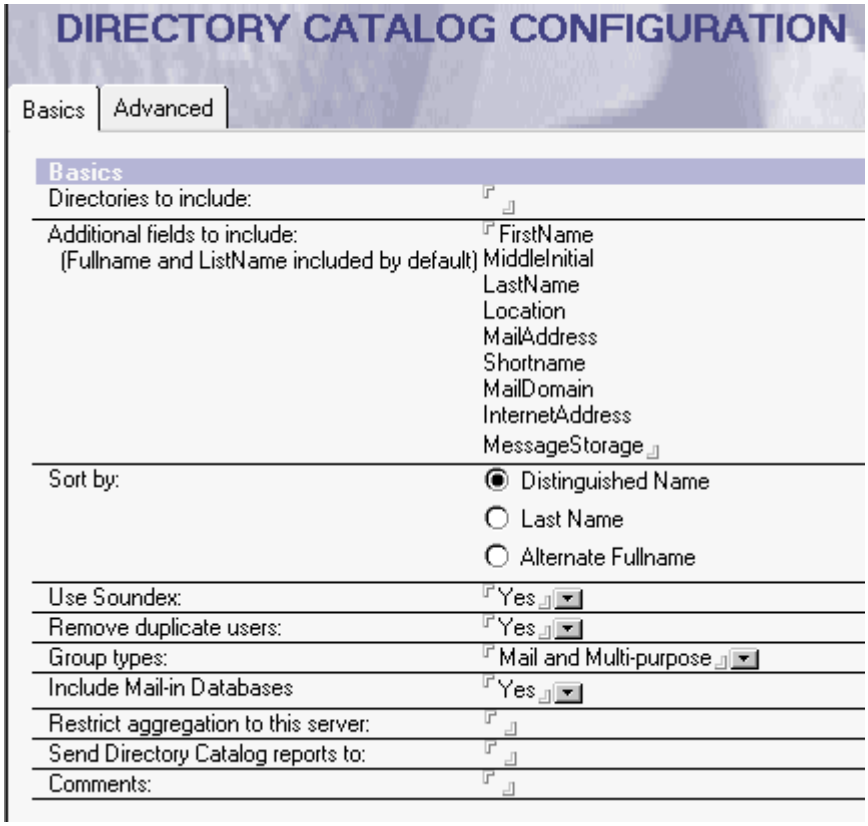

Figure 66. Directory catalog configuration: basics

In the Directories to include field, list the file names of the directories that you want to aggregate into this directory catalog. It is strongly recommended that you have replicas of each directory on this server. If not, you can use the syntax `port!!!server!!filename` to include directories on foreign servers, for example `tcpip!!!hubserver1/acme!!acmedir.nsf`.

The next field controls what fields from the Domino directories are added to the directory catalog. The directory catalog includes FullName (for people) and ListName (for Groups) by default. If you want to include any other information, make sure that the field containing that information is included in the configuration. One field you probably do want to include, at least for server-based directory catalogs, is the Members field from group documents. This will allow the server to retrieve the group's member list from the directory catalog rather than the source directory.

Once you have made all of your configuration settings, save and close the configuration document. You are now ready to build the Directory Catalog.

3.5.6.1 Running the directory catalog aggregator

On the server with the directory catalog database, type `load dircat <filename>` on the server console. This will begin the process of building the directory catalog. Once the directory catalog is built, you should update the full text index by typing `load updall <filename>` from the server console. The directory catalog makes extensive use of the full-text index, so don't forget this step.

3.5.7 Making the directory catalog available to the server

The final step in setting up a server-based directory catalog is to make it available to the server. If you are deploying this directory catalog onto every server in your Domino environment, you should edit the Domino directory profile. To do this, open the Domino directory and choose **Actions -> Edit Directory Profile**. In the directory catalog database name for domain field, type the name of the directory catalog.

If you want to set up the directory catalog only on certain servers, or if you want to use different directory catalogs on different servers, you should edit the individual server documents. To do this, use the Domino Administrator client. First, make sure that the server you want to use the directory catalog is chosen, then choose the **Configuration** tab, expand the **Server** section, then click **Current Server Document**. Click the **Edit Server** action button to make your changes (Figure 67). Now move the cursor down to the directory catalog database name on this server field and type in the filename of the directory catalog. Click **Save** and **Close**, then reboot the server to have this change take effect.

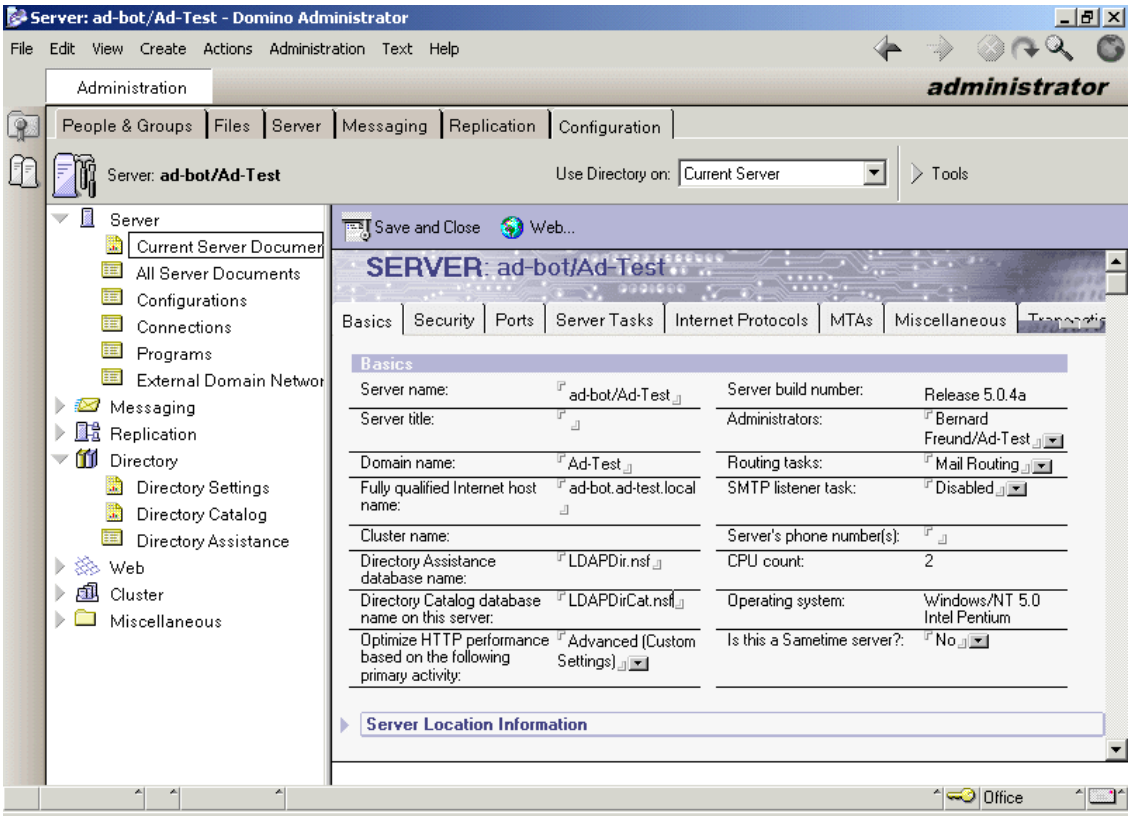

Figure 67. Deploying the directory catalog

3.6 Bringing it all together

After we had the Domino directory, Active Directory and the SecureWay Directory correctly configured, the final task was to simulate a client doing a search for data on each directory service, while accessing the other.

Since we already tested Active Directory and SecureWay referrals in Chapter 2, “Scenario1: Integrating SecureWay with Active Directory” on page 45, we focused on the Domino elements added in this chapter.

3.6.1 Testing the Domino directory basic LDAP functionality

Our first step was to test the basic LDAP search capabilities of the Domino directory, as we did with the Active Directory and the IBM SecureWay Directory in the previous chapter.

Note

For the sake of simplicity, we assumed that you already read Chapter 2, “Scenario1: Integrating SecureWay with Active Directory” on page 45. We repeated the main procedures in this section, for your convenience, but for more detailed instructions on the basic elements, such as starting the necessary programs and non-essential operations, please refer to 2.6, “Bringing it all together” on page 60.

1. The first step is clicking **Start**, then **Run**, then pointing to the path to the LDP utility executable (for directions on how to install LDP and other utilities, please refer to Appendix B, “Client tools” on page 155). The LDP utility will start.
2. The next step is to establish a session to the LDAP (Domino) server. To do this, click **Connection**, and then **Connect**.
3. In the Connect dialog box (Figure 68), type the name or IP address and the port of the target LDAP server. Click **OK**.

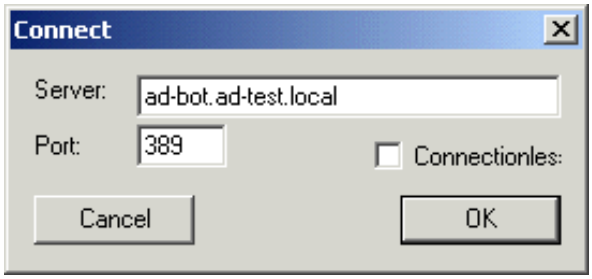

Figure 68. Connecting to Domino

4. You can see part of the Domino directory RootDSE in Figure 69. The right-hand side of the window shows the standard LDAP V3 RootDSE, plus additional attributes. For a list of the RootDSE standard attributes, please see the LDAP V3 RFC and Appendix A, “The RootDSE” on page 153. For a list of relevant LDAP RFCs, please see Appendix D, “Standards” on page 175.

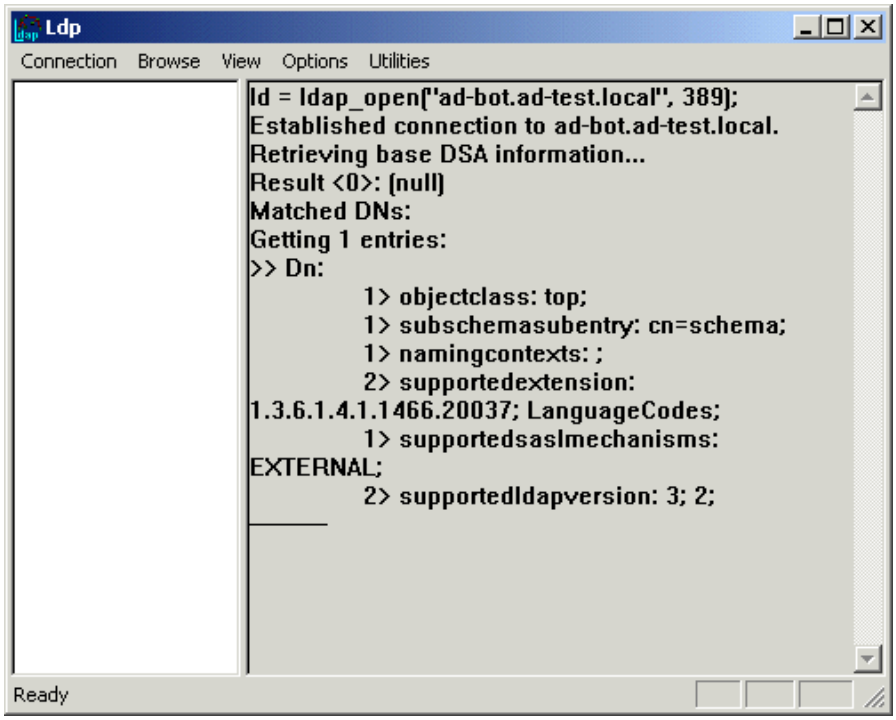

Figure 69. The Domino directory RootDSE

5. Now we are ready to perform the search. To accomplish this, we can either right-click the (optional) tree view and select **Search**, or select **Browse**, and then **Search**.
6. In the Search dialog box that follows (Figure 70), you should type the base DN (the starting point) for this search and the criteria for this search, such as `objecttype=*` or `cn=admin*`. You should also select the scope: **Base** (just the base DN), **One Level** (the base DN plus its immediate children) or **Subtree** (everything under the base DN, no matter how many levels deep).

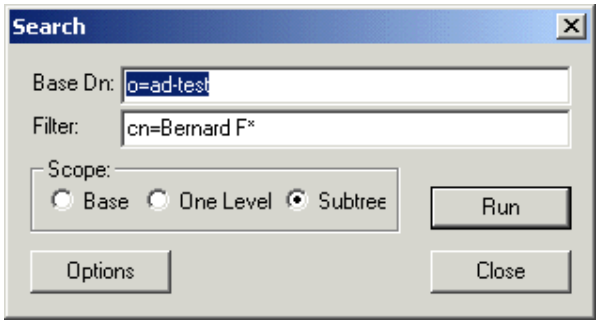

Figure 70. Searching for Domino data

If you have a very large directory, you may want to click the **Options** button and set time and size limits, choose a different search call type, or disable the **Chase referrals** option in the Search Options dialog.

7. You can see the results of the search below:

```
***Searching...
ldap_search_s(ld, "o=ad-test", 2, "cn=Bernard*", attrList,  0, &msg)
Result <0>: (null)
Matched DNs:
Getting 1 entries:
>> Dn: CN=Bernard Freund,O=Ad-Test
    5> objectclass: top; person; organizationalPerson; inetOrgPerson; dominoPerson;
    1> cn: Bernard Freund;
```

3.6.2 Searching for SecureWay data through Domino

The next step was to test the referrals to the enterprise directory (SecureWay). This means that the client would connect to the Domino directory server and query for data stored on the SecureWay Directory.

To evaluate and test this case, we executed all steps from 3.6.1, “Testing the Domino directory basic LDAP functionality” on page 97, but instead of specifying a base DN that belongs to the naming context stored on the Domino directory server, we specified a base DN belonging to the naming context stored on the SecureWay Directory, as shown in Figure 71.

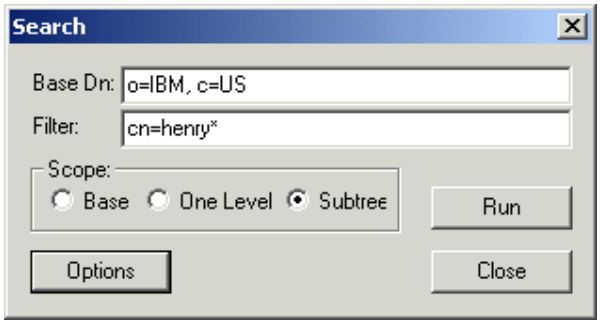

Figure 71. Searching for SecureWay data while connected to Domino

The results you receive will depend on whether the **Chase Referrals** option is enabled in the Search Options dialog box. In our sample search, the result received when the option was disabled was the actual referral:

```
***Searching...
ldap_search_s(ld, "o=IBM, c=us", 2, "cn=henry*", NULL,  0, &msg)
Error: Search: Referral. <10>
Result <10>:
Matched DNs:
Getting 0 entries:
```

When this option is enabled, on the other hand, the client application will chase the referral and fetch the data correctly:

```
***Searching...
ldap_search_s(ld, "o=IBM, c=us", 2, "cn=henry*", NULL,  0, &msg)
Result <0>: (null)
Matched DNs:
Getting 1 entries:
>> Dn: cn=Henry Nguyen, ou=Austin, o=IBM, c=US
    3> objectclass: person; organizationalPerson; top;
    1> cn: Henry Nguyen;
    1> sn: Nguyen;
    1> telephonenumber: 1-812-855-4028;
    1> internationalisdnnumber: 755-4028;
    2> facsimiletelephonenumber: 1-812-855-9087; 755-9087;
    1> title: AIX Support;
    1> postalcode: 9551;
```

Note

Throughout this section, we have used disjointed namespaces. It is also possible to use a contiguous namespace in both directories, and use the same base DN for all queries.

3.6.3 Searching for Domino data through SecureWay

The third case we wanted to test was referrals to the Domino directory. This means that the client would connect to a SecureWay Directory server and query for data stored in the Domino directory.

To evaluate and test this case, we executed all steps from 2.6.3, “Searching for data in the SecureWay Directory” on page 72, but instead of specifying a base DN that belongs to the naming context stored on the SecureWay server, we specified a base DN on the Domino naming context.

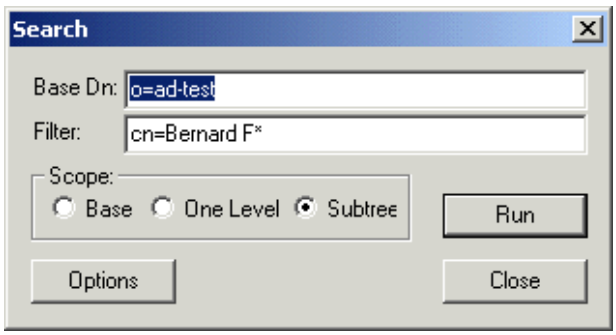

Figure 72. Searching for Domino directory data from SecureWay

The results you will receive will depend on whether the **Chase Referrals** option is enabled in the Search Options dialog box. On our sample search, the result received when the option was disabled was the actual referral:

```
***Searching...
ldap_search_s(ld, "o=ad-test", 2, "cn=Bernard F*", NULL,  0, &msg)
Error: Search: Referral. <10>
Result <10>:
Matched DNs:
Getting 0 entries:
```

When this option is enabled, on the other hand, the client application will chase the referral and fetch the data correctly:

```
***Searching...
ldap_search_s(ld, "o=ad-test", 2, "cn=Bernard F*", NULL,  0, &msg)
Result <0>: (null)
Matched DNs:
Getting 1 entries:
>> Dn: CN=Bernard Freund,O=Ad-Test
   1> cn: Bernard Freund;
   1> shortname: bfreund;
   1> uid: bfreund;
   1> mail: Bernard_Freund/Ad-Test%Ad-Test@ad-test.local;
   5> objectclass: top; person; organizationalPerson; inetOrgPerson;
dominoPerson;
   1> certificate: 03003302 3AE5A03E 09G0161B G002D189;
```

3.6.4 Searching for Domino data through Active Directory

Now we wanted to test referrals to servers two levels removed from our connection point. This means that the client would connect to an Active Directory server and query for data stored on the Domino directory.

To evaluate and test this case, we executed all steps from 3.6.3, “Searching for Domino data through SecureWay” on page 102, but instead of connecting directly to the IBM SecureWay Directory server, we connected to the Active Directory server.

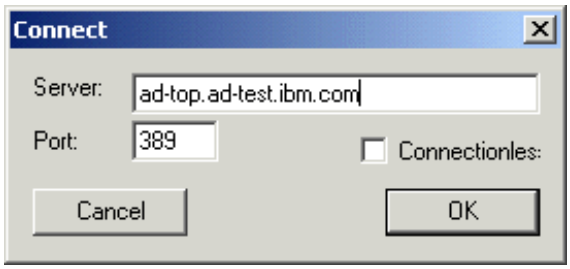

Figure 73. Searching for Domino directory data from Active Directory

The results you will receive will depend on whether the **Chase Referrals** option is enabled in the Search Options dialog box. On our sample search, the result received when the option was disabled was the actual referral:

```
***Searching...
ldap_search_s(ld, "o=ad-test", 2, "cn=Bernard F*", NULL,  0, &msg)
Error: Search: Referral. <10>
Result <10>: 0000202B: RefErr: DSID-031005EE, data 0, 1 access points
    ref 1: 'ldap1.ad-test.ibm.com'
Matched DNs:
Getting 0 entries:
```

When this option is enabled, on the other hand, the client application will chase the referral and fetch the data correctly:

```
***Searching...
ldap_search_s(ld, "o=ad-test", 2, "cn=Bernard F*", NULL,  0, &msg)
Result <0>: (null)
Matched DNs:
Getting 1 entries:
>> Dn: CN=Bernard Freund,O=Ad-Test
   1> cn: Bernard Freund;
   1> shortname: bfreund;
   1> uid: bfreund;
   1> mail: Bernard_Freund/Ad-Test%Ad-Test@ad-test.local;
   5> objectclass: top; person; organizationalPerson; inetOrgPerson;
dominoPerson;
   1> certificate: 03003302 3AE5A03E 09G0161B G002D189;
```

3.6.5 Searching for Active Directory data through Domino

The final step was to test referrals to servers two levels removed from our connection point, but this time initiating the search from the Domino side. This means that the client would connect to a Domino directory server and query for data stored in the Active directory.

Note

Whenever you use referrals in Domino, you should not forget that Domino *always* searches its own directory, including secondary Domino directories, before handing out referrals. So, if any records in the Domino directory match the search criteria, no referrals will be returned to the client application. This holds true even if you specify a base DN outside the Domino namespace in your LDAP query.

To evaluate and test this case, we executed all steps from 3.6.3, “Searching for Domino data through SecureWay” on page 102, but instead of connecting

directly to the IBM SecureWay Directory server, we connected to the Domino server.

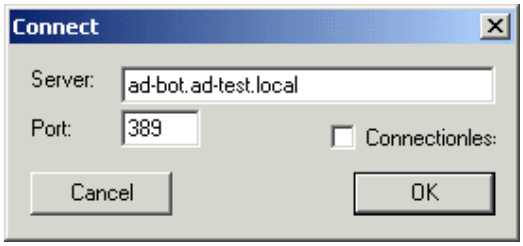

Figure 74. Searching for Active Directory data from Domino

The results you will receive will depend on whether the **Chase Referrals** option is enabled in the Search Options dialog box. In our sample search, the result received when the option was disabled was the actual referral:

```
***Searching...
ldap_search_s(ld, "dc=com", 2, "cn=harri*", NULL,  0, &msg)
Error: Search: Referral. <10>
Result <10>:
Matched DNs:
Getting 0 entries:
```

When this option is enabled, on the other hand, the client application will chase the referral and fetch the data correctly:

```
***Searching...
ldap_search_s(ld, "dc=com", 2, "cn=harri*", NULL,  0, &msg)
Result <0>: (null)
Matched DNs:
Getting 1 entries:
>> Dn: CN=Harri Levanen,CN=Users,DC=ad-test,DC=ibm,DC=com
   1> accountExpires: 9223372036854775807;
   1> badPasswordTime: 0;
   1> badPwdCount: 0;
   1> codePage: 0;
   1> cn: Harri Levanen;
   1> countryCode: 0;
   1> displayName: Harri Levanen;
   1> givenName: Harri;
   1> instanceType: 4;
   1> lastLogoff: 0;
   1> lastLogon: 0;
   1> logonCount: 0;
   1> distinguishedName: CN=Harri
Levanen,CN=Users,DC=ad-test,DC=ibm,DC=com;
   1> objectCategory:
CN=Person,CN=Schema,CN=Configuration,DC=ad-test,DC=ibm,DC=com;
   4> objectClass: top; person; organizationalPerson; user;
   1> objectGUID: 7465410a-bc09-4732-b5b5-ab6ca32b48d3;
   1> objectSid: S-15-65D637A8-AB48F72-462B3D83-45A;
   1> primaryGroupID: 513;
   1> pwdLastSet: 126164620803125000;
   1> name: Harri Levanen;
   1> sAMAccountName: harri;
   1> sAMAccountType: 805306368;
   1> sn: Levanen;
   1> userAccountControl: 66048;
```

3.7 What we have discovered

Once the IBM SecureWay Directory, the Domino directory and Active Directory were properly configured and set up to do referrals, we were able to successfully search for data on either directory no matter which server we connected to.

Therefore, in our fictional scenario, referrals make it possible to have a very simple (albeit somewhat primitive) form of enterprise directory, depending on the applications' needs, of course. The IBM SecureWay Directory successfully acted as the hub for this configuration, proving it is a very open and flexible directory solution.

Chapter 4. Scenario3: Integrating LDAP Server and Active Directory

The following scenario is of a fictional company that keeps its enterprise directory in the LDAP server running on a S/390 mainframe. This company only recently defined its policies towards the use of client/server environments. While there was no corporate policy in place, one division, attending to an internal need, created a Windows 2000 domain, and it stored several types of business data in the Active Directory. In this scenario we will explore referral functionality by having an LDAP client searching for data stored in a directory (the enterprise directory) by connecting to another directory (Active Directory) and vice versa.

Note

IBM's LDAP product is named SecureWay Directory on supported platforms with the exception of the OS/390. On OS/390 the LDAP product is a part of SecureWay Security Server and is named LDAP Server.

4.1 About the environment

Table 4 is a listing of the environment.

Table 4. Specifications for the second scenario

Machine DNS Name	MVS03A	AD-TOP.AD-TEST.IBM.COM
Operating system	OS/390 V2R10	Windows 2000 Advanced Server SP1
Domain	(none)	ad-test.ibm.com
Domain role	N/A	domain controller
Additional software	UNIX System Services	(none)
	CS for OS/390 IP Services	
	DB2 V5.1	
Additional OS components	SecureWay Security Server -RACF -LDAP Server	Microsoft IIS V5.0

The scenario environment is illustrated in Figure 75.

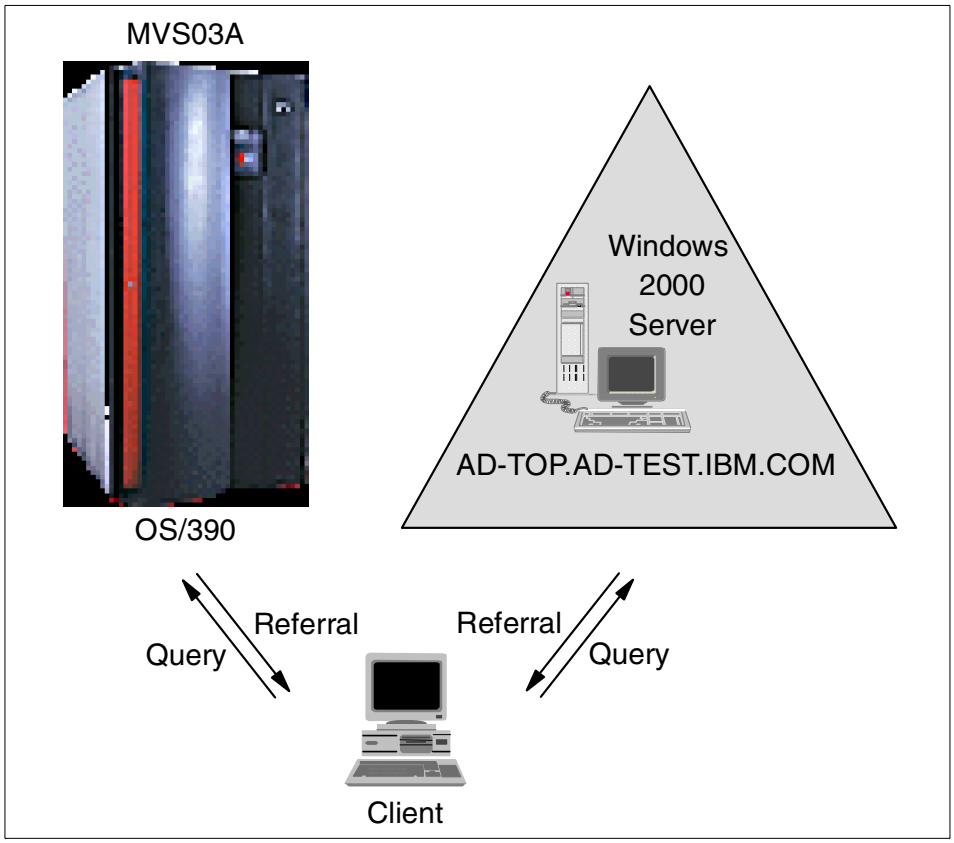

Figure 75. OS/390 scenario environment

4.2 Configuring LDAP Server

For instructions on installation and configuration of LDAP Server refer to *OS/390 Security Server 1999 Updates: Installation Updates*, SG24-5629, and *OS/390 Security Server LDAP Server Administration and Usage Guide*, SC24-5861.

For our scenario we used a domain suffix of `o=IBM_US,c=US` for the LDAP Server. See the sample environment section in Appendix F.1, “Sample started procedure” on page 181.

4.3 Setting up LDAP Server for referrals

The entry

```
referral ldap://ad-top.ad-test.ibm.com:389/dc=ad-test,dc=ibm,dc=com
```

was added to the configuration file to add a referral to Active Directory. See the sample configuration file in Appendix F.3, “Sample configuration file” on page 184.

4.4 Setting up an Active Directory referral

A referral was created in Active Directory by adding a naming context name of `o=IBM_US,c=US`, which is the suffix of the directory on OS/390. You may reference 2.5, “Creating a referral in Active Directory” on page 50 for information on adding the referral to the Active Directory.

4.5 Bringing it all together

The final task for this scenario was to simulate a client doing a search for data on each directory service, while accessing the other.

We analyzed two possible cases:

1. A client connects to an Active Directory server, searches for data stored in the LDAP Server, receives a referral, and chases this referral.
2. A client connects to the LDAP Server, searches for data stored in an Active Directory server, receives a referral, and chases this referral.

4.5.1 Searching for data in the LDAP Server from Active Directory

The simplest way to search for data in the LDAP Server is when the client connects to the LDAP Server to make the query. In this case, we will connect to the Active Directory Server and search for data in the LDAP Server, on OS/390, through a referral.

1. The first step is to start the LDP utility by clicking **Start**, then **Run**, then pointing to the path to the LDP utility executable (for directions on how to install LDP and other utilities, please refer to Appendix B, “Client tools” on page 155).

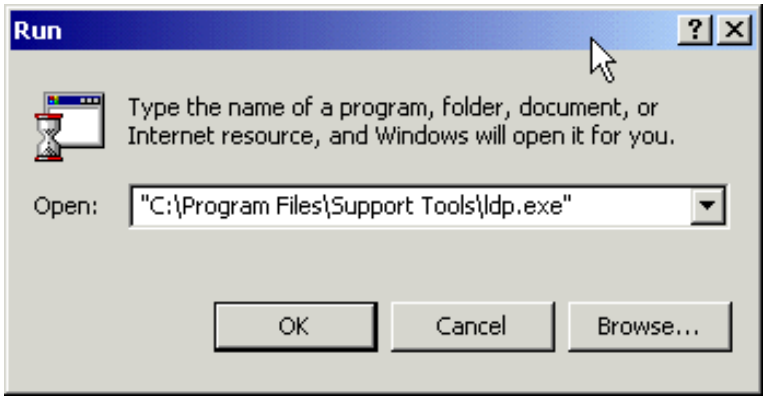

Figure 76. Starting the LDP utility

2. The LDP utility will start, as shown in Figure 77.

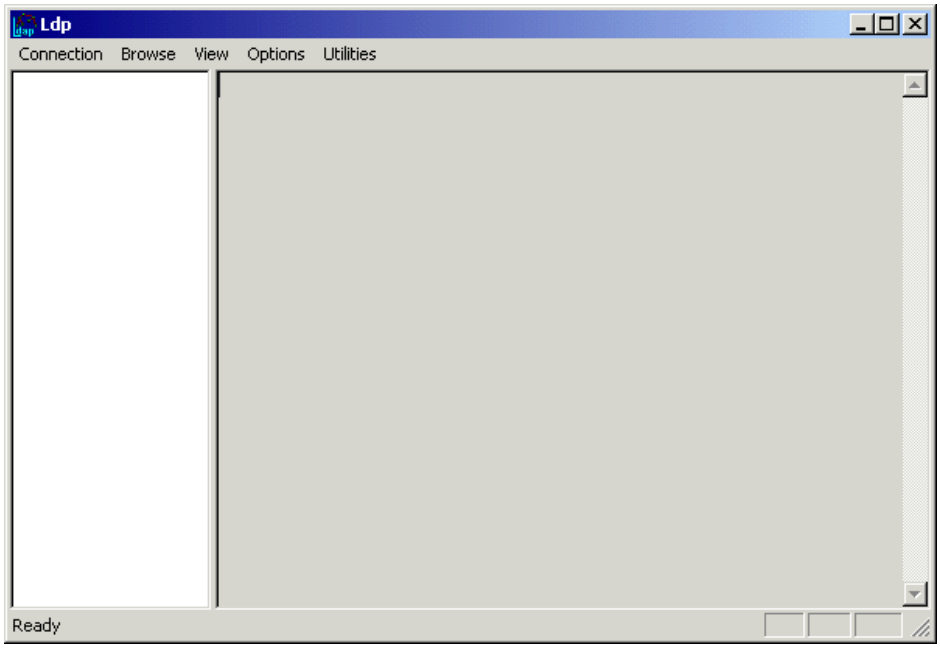

Figure 77. The LDP utility

3. The next step is to establish a session to the Active Directory server. To do this, click **Connection**, and then **Connect**, as shown in Figure 78.

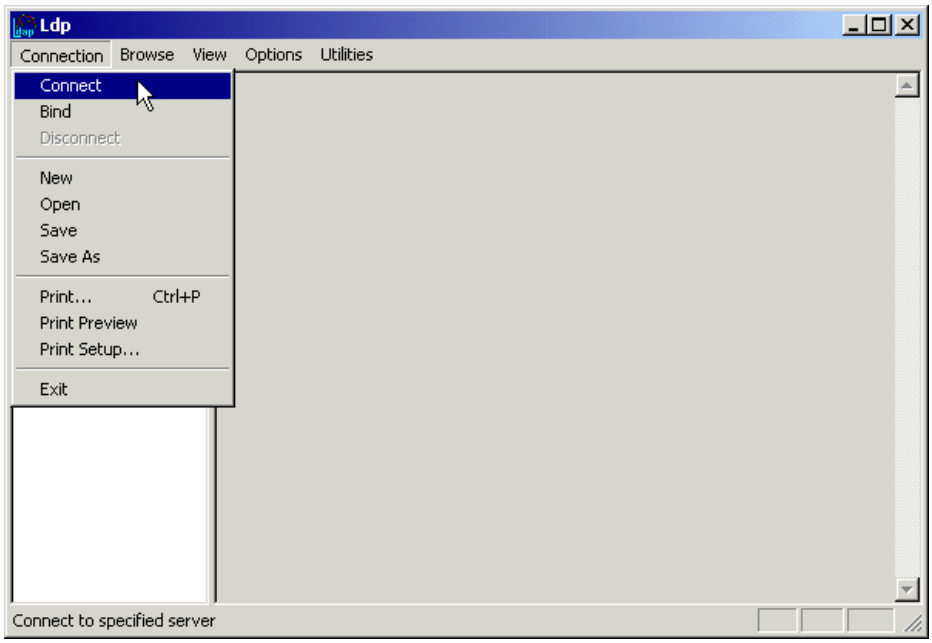

Figure 78. Establishing a session to the LDAP server

4. In the Connect dialog box (Figure 79), type the name or IP address and the port number of the directory server to connect to. Click **OK**.

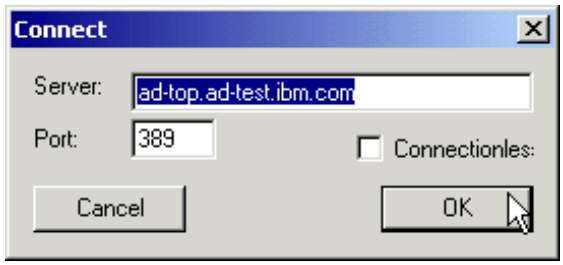

Figure 79. The connect dialog box

5. You can see part of the Active Directory RootDSE in Figure 80. The right-hand side of the window shows the standard LDAP v3 RootDSE, plus additional attributes. For a list of the RootDSE standard attributes, please see the LDAP V3 RFC. For a list of relevant LDAP RFCs, please see Appendix D, “Standards” on page 175.

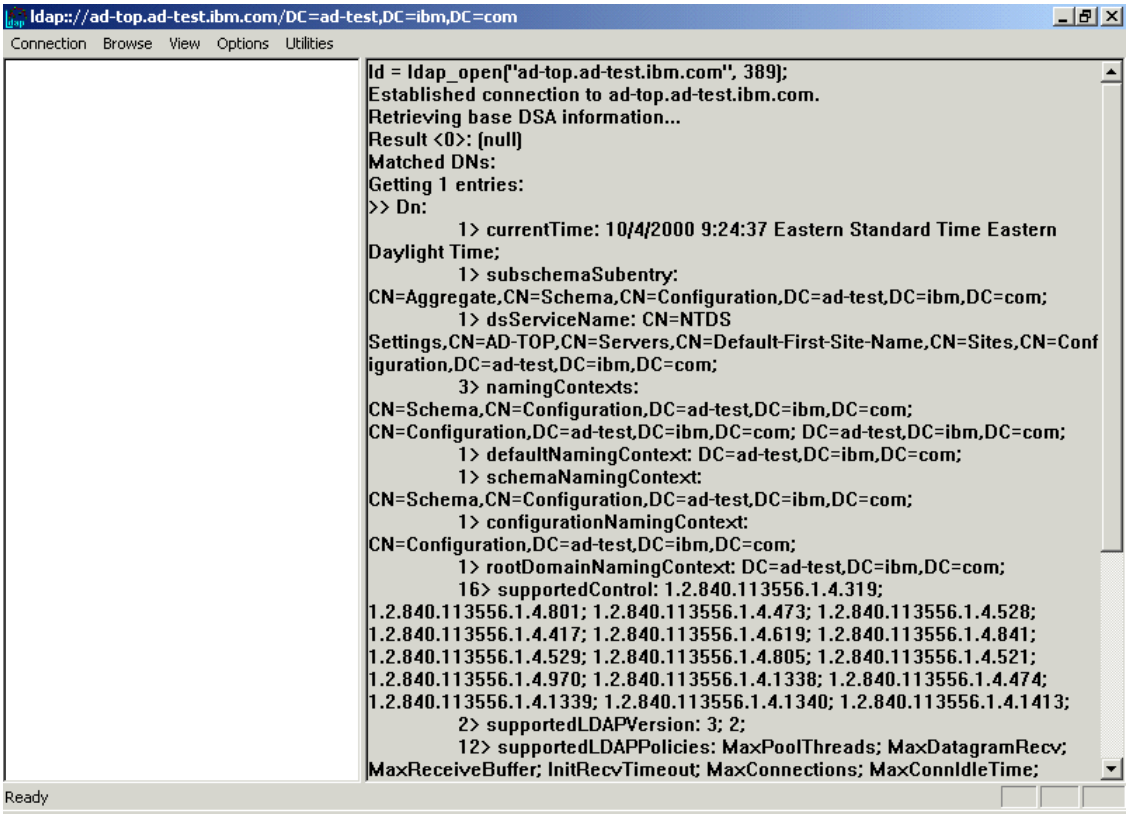

Figure 80. The Active Directory RootDSE

6. The final step you may want to perform before actually executing the search is to display a tree view on the left-hand side of the window. This is actually an optional step, but it simplifies the query process and it also reduces typing (and the chance of making typing mistakes). To display a tree view, select **View**, and then **Tree**, as shown in Figure 81.

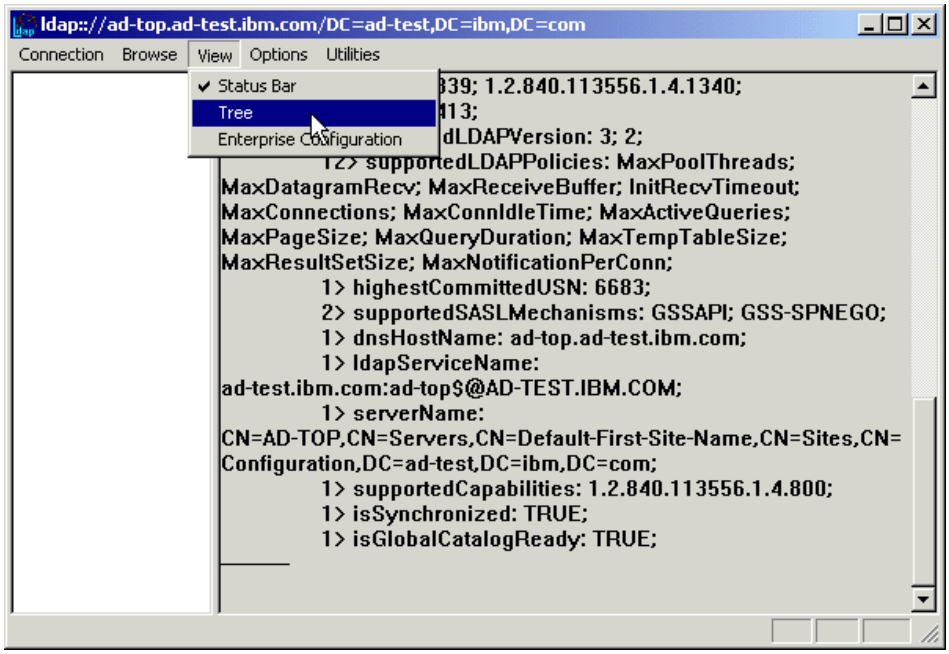

Figure 81. Setting up the tree view

7. In the Tree View dialog box (Figure 82), you should type the common name of the naming context you want to display in the tree view. In this case, we wanted the default user naming context.

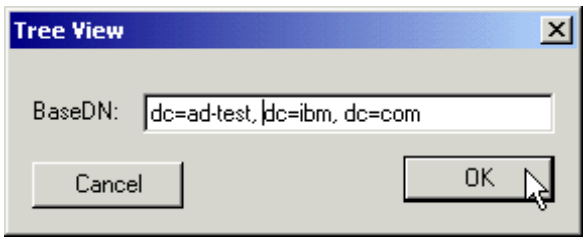

Figure 82. Choosing the naming context to be displayed in the tree view

8. The tree view will be displayed on the right-hand side of the window, as shown in Figure 83.

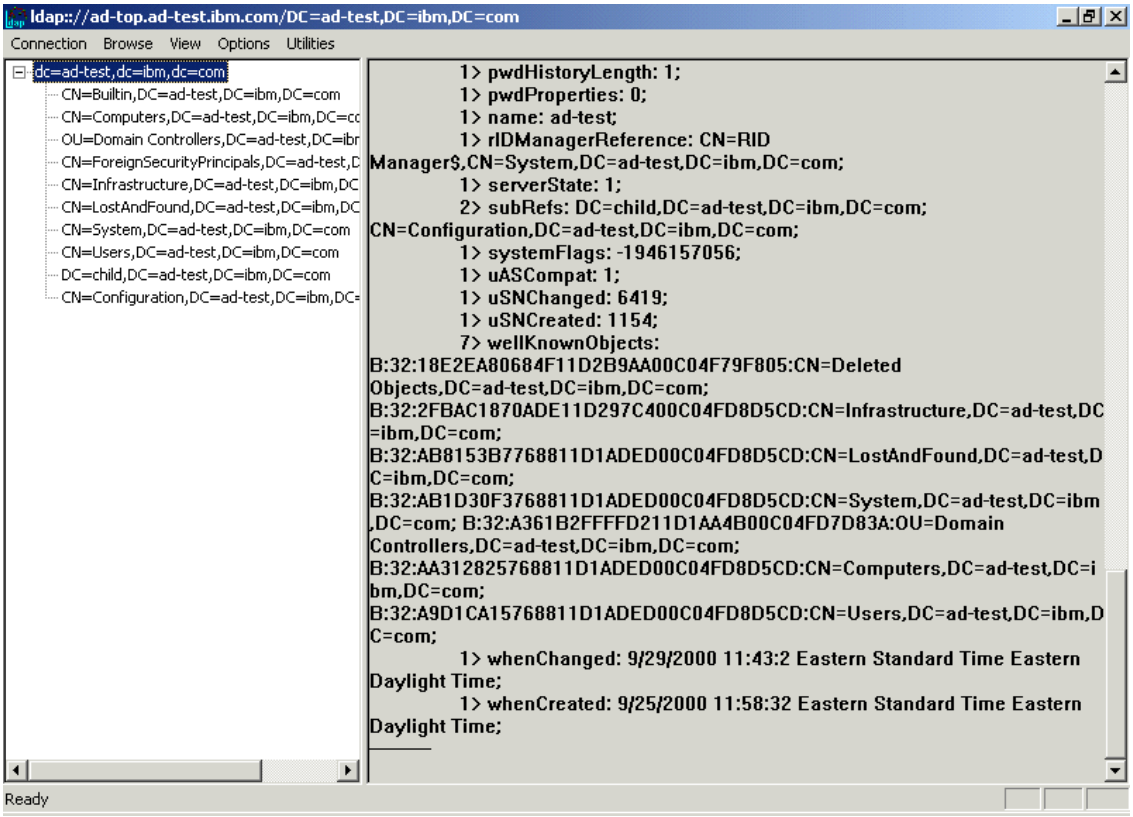

Figure 83. The tree view

9. Finally, we are ready to perform the search. To accomplish this, we can either right-click the tree view and select **Search**, or select **Browse**, and then **Search**, as shown in Figure 84.

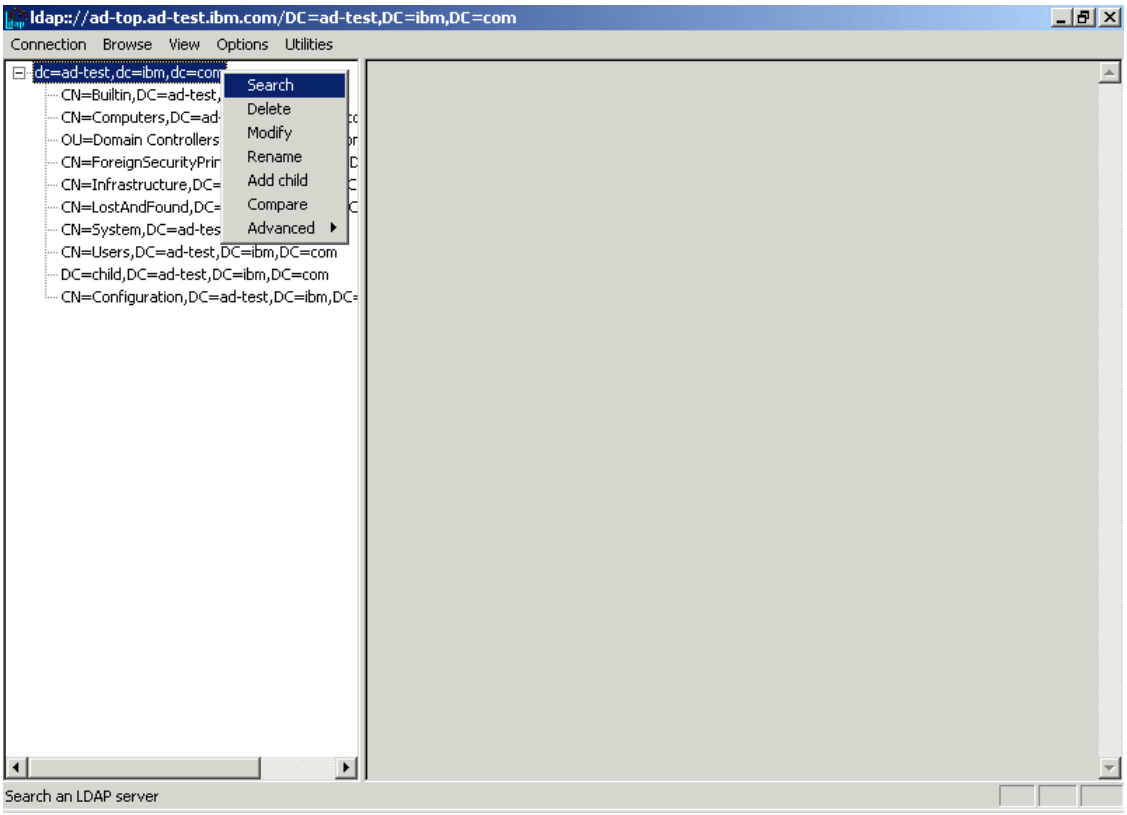

Figure 84. Initiating a search

10. In the Search dialog box that follows (Figure 85), you should type the base DN (the starting point) of the LDAP Server naming context in the Search dialog box.

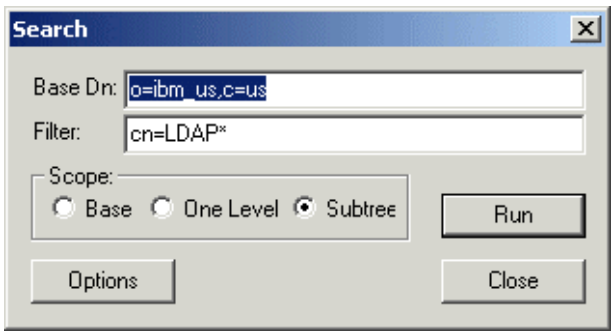

Figure 85. Searching the LDAP Server

11.The results of this search can be seen in Figure 86

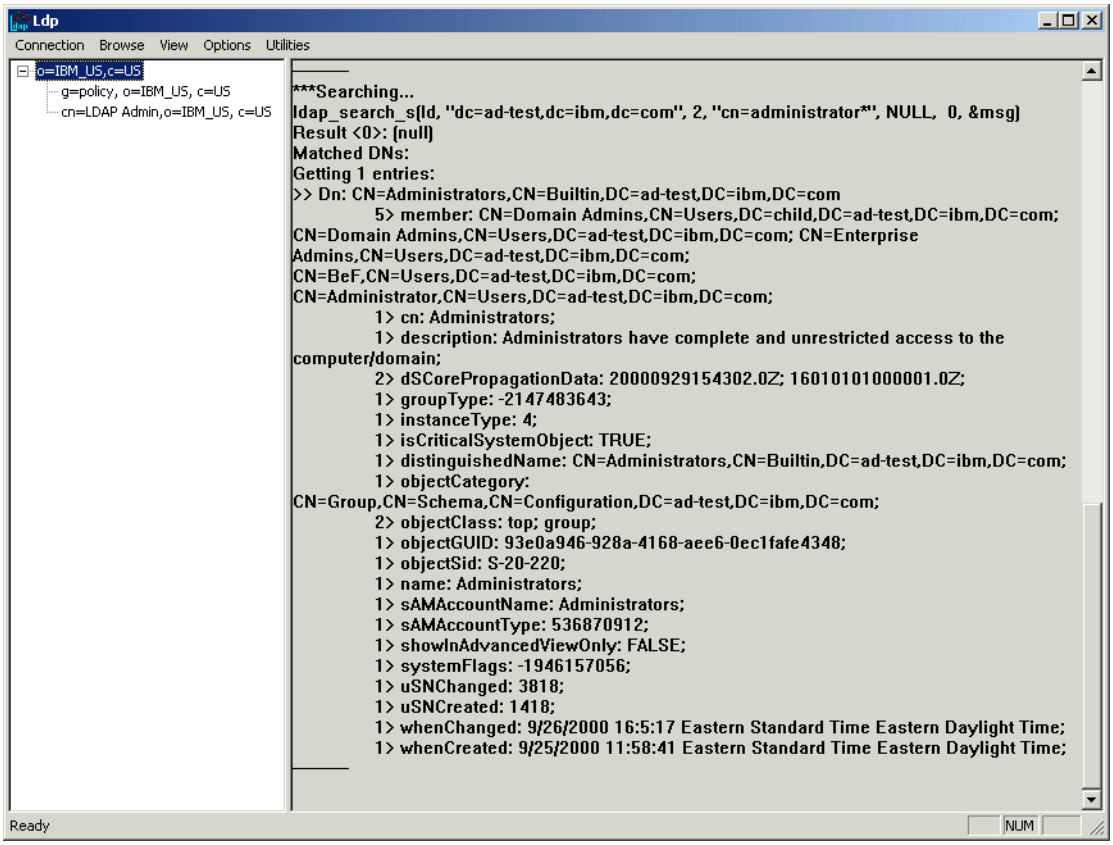

Figure 86. LDAP Server search results

4.5.2 Searching for data in Active Directory from the LDAP Server

The simplest way to search for data in Active Directory is when the client connects to the Active Directory server to make the query. In this case, we will connect to the LDAP Server on OS/390 and search for data in Active directory, through a referral.

In this case we connected to the LDAP Server on OS/390 with the LDP utility referenced in Appendix B, “Client tools” on page 155.

To evaluate and test this case, we executed all steps from 4.5.1, “Searching for data in the LDAP Server from Active Directory” on page 109, with the following differences:

- In step 4 on page 111, we supplied the address to the LDAP Server. This is shown in Figure 87.

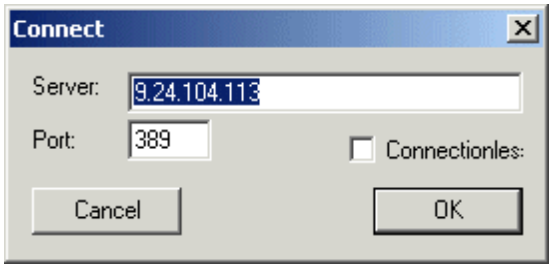

Figure 87. Connecting to LDAP Server

This produced the following RootDSE:

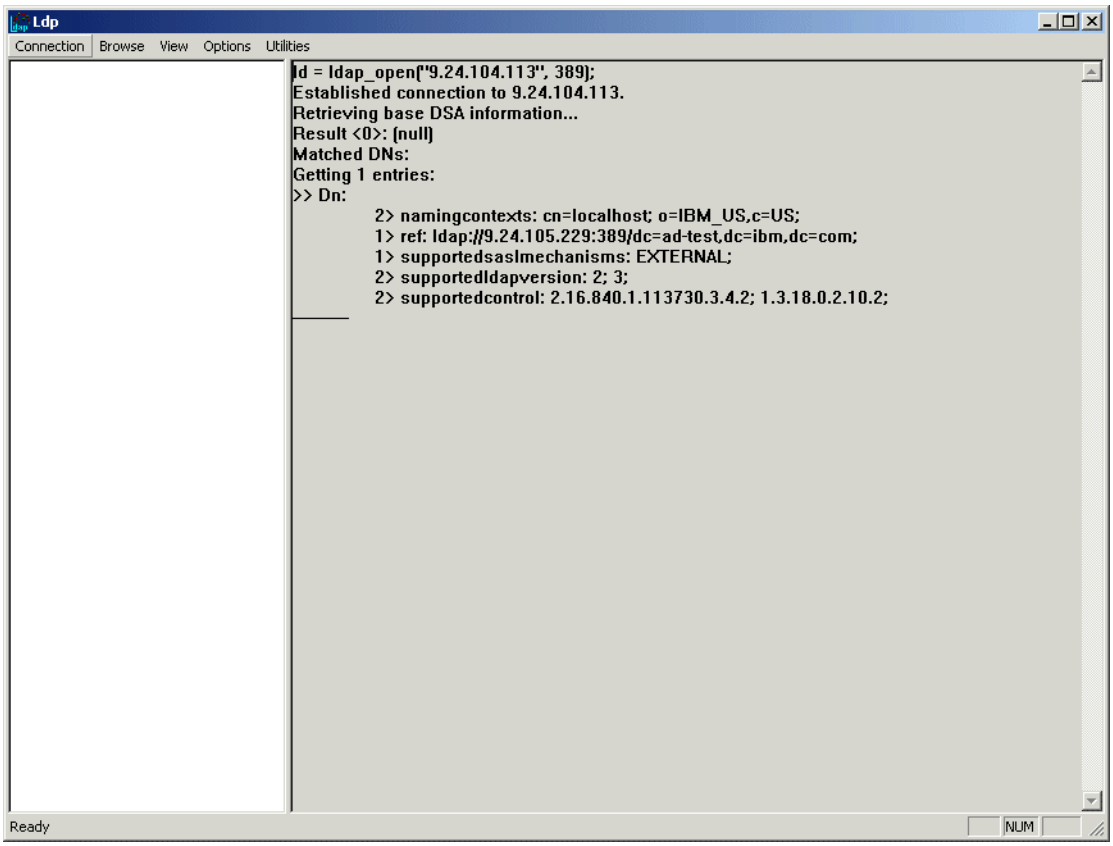

Figure 88. The LDAP Server RootDSE

- In step 7 on page 113, we supplied the naming context stored in the LDAP Server. This is shown in Figure 89.

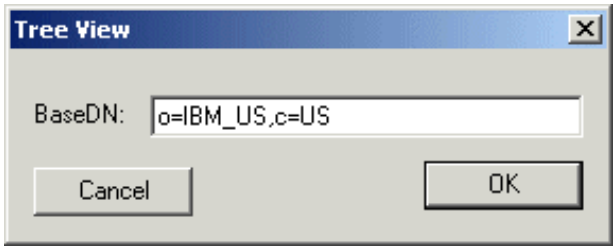

Figure 89. The LDAP Server naming context

These actions produced the results displayed in Figure 90.

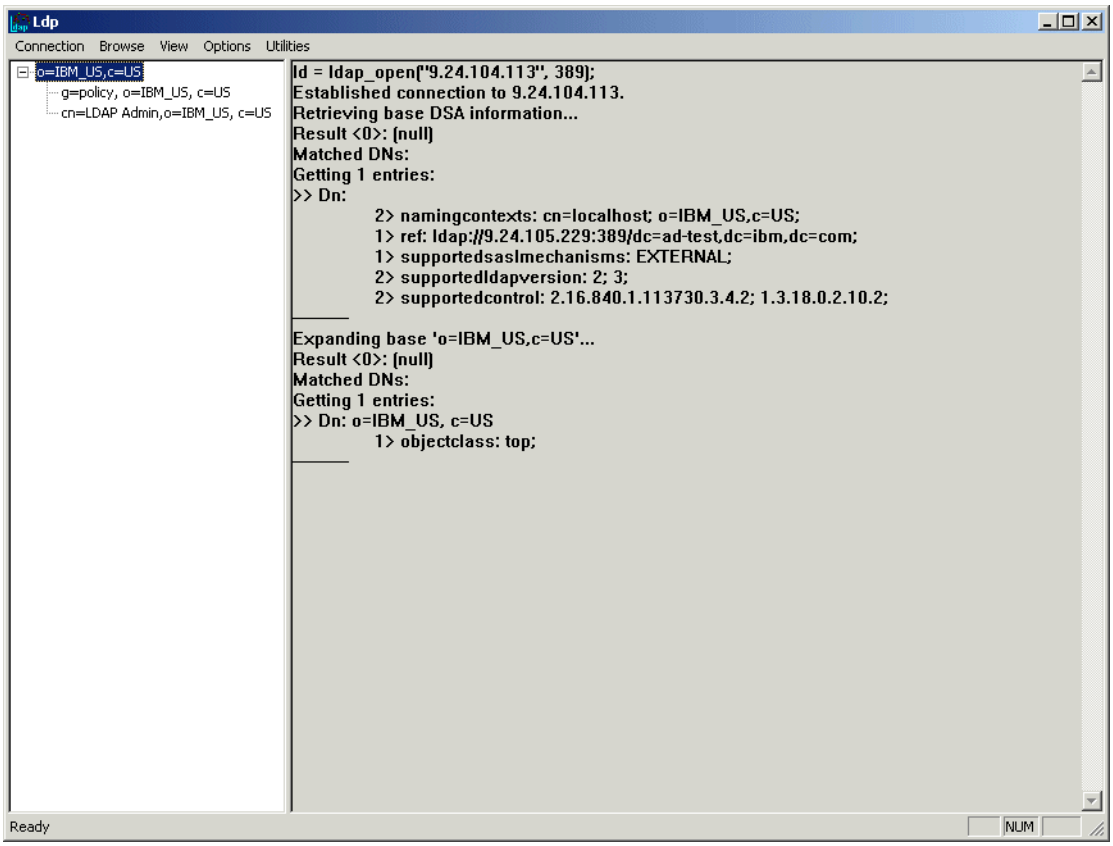

Figure 90. The LDAP tree view

- Finally, in step 10 on page 115, we supplied the base DN of the LDAP Server naming context in the Search dialog box (Figure 91).

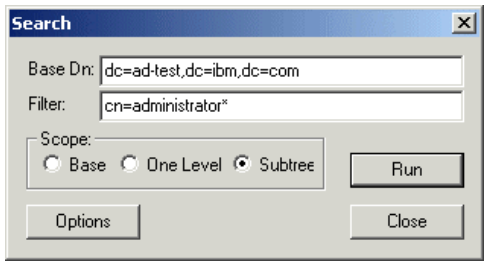

Figure 91. Searching for data in Active Directory through the LDAP Server

The results of this search can be seen in Figure 92.

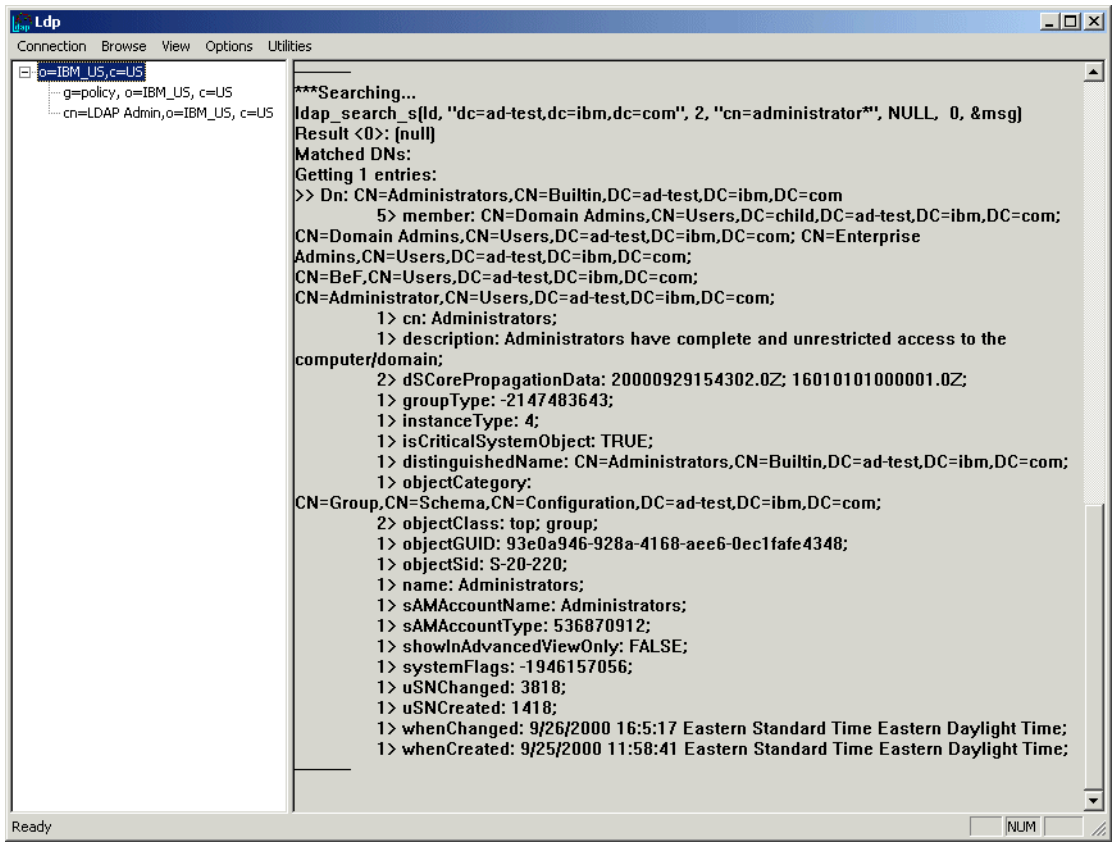

Figure 92. LDAP Server search results

4.6 What we discovered

Once the LDAP Server and Active Directory were properly configured and set up to do referrals, we were able to successfully search for data in either directory no matter which server we connected to. Therefore, referrals make it possible to have SecureWay Directory and Active Directory co-exist in the same environment.

In our fictional scenario, this means the company could use this functionality to enable a limited coexistence for the company's applications, while the Active Directory data for this division is migrated to the enterprise directory (LDAP Server).

Chapter 5. Scenario4: Extending schemas and synchronizing data

The following scenario is of a fictional company that has defined the IBM SecureWay Directory as its enterprise directory. This company also has a Windows 2000 domain. The company wants to administer the data in one directory and have the updates ported to the other directory, in order to be cost efficient and have accurate data maintained between the directories.

In this scenario we will explore the extensibility of the schema and the exchange of data between the SecureWay Directory and Active Directory.

For the scenario we will create a new class, ldapPerson, in both directories, and instances of that class in the SecureWay Directory. Then we will copy the instances to Active Directory through the use of an LDIF file.

5.1 About our test environment

To test this exchange of data we decided to create a common class between the two directories. Therefore, we created the ldapPerson class, a sub-class of the Person class, in both directories. The common class requires common mandatory attributes; see Table 5 on page 121. Then we created two instances of the ldapPerson class in the SecureWay Directory, which were then exported to an LDIF file. The LDIF file was then imported to the Active Directory.

Table 5. Listing of the newly added class

Directory	Class	Attributes (attributes for the ldapPerson class)
SecureWay Directory	ldapPerson	cn (inherited from the Person class)
		sn (inherited from the Person class)
		objectClass (inherited from the top class)
		objectCategory (inherited from the top class)
		instanceType (*manually add*)
		nTSecurityDescriptor (*manually add*)

Directory	Class	Attributes (attributes for the ldapPerson class)
Active Directory	ldapPerson	cn (inherited from the Person class)
		sn (inherited from the Person class)
		objectClass (inherited from the top class)
		instanceType(inherted from the top class)
		nTSecurityDescriptor (inherited from the top class)
		objectCategory (inherited from the top class)

Note

The use of the word *common* in this chapter refers either to attributes or classes that exist in both SecureWay Directory and Active Directory.

5.2 Creating attributes for our scenario

For the ldapPerson class to have a common set of attributes the following attributes were added to the SecureWay Directory: objectCategory, nTSecurityDescriptor, and instanceType.

5.2.1 Creating new attributes in SecureWay

The following is a sample of adding an attribute, objectCategory, to the SecureWay Directory using the IBM SecureWay Directory Management Tool. To add an attribute select the **Add** attribute button and enter the attribute name, OID, and syntax.

Note

We used the same OID numbers for each of the attributes in the SecureWay Directory as in the Active Directory.

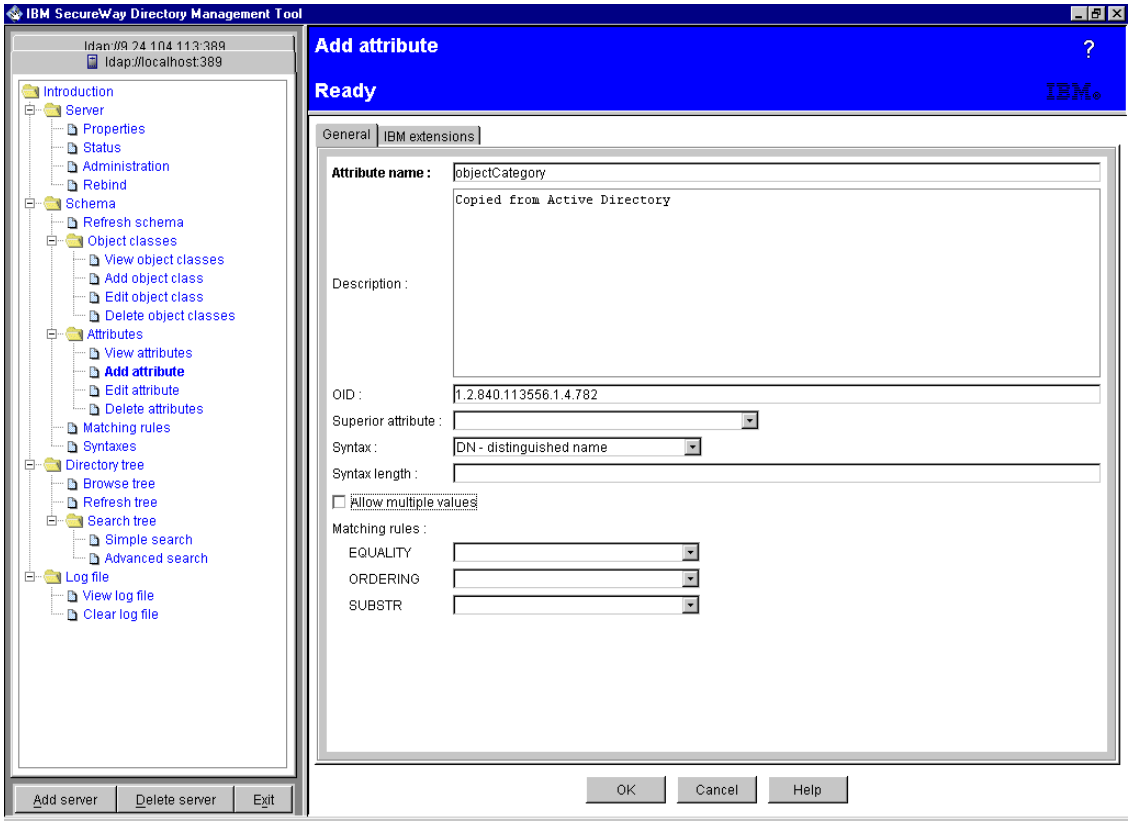

Figure 93. SecureWay - adding an attribute

5.2.2 Creating new attributes in Active Directory

We did not need to add any attributes to Active Directory; see Table 5 on page 121.

5.3 Creating classes for our scenario

The following shows how to create the ldapPerson class for both SecureWay and Active Directory.

5.3.1 Creating new classes SecureWay

Using the IBM SecureWay Directory Management Tool we created an ldapPerson class by selecting **View object class** then **Add**; see Figure 94.

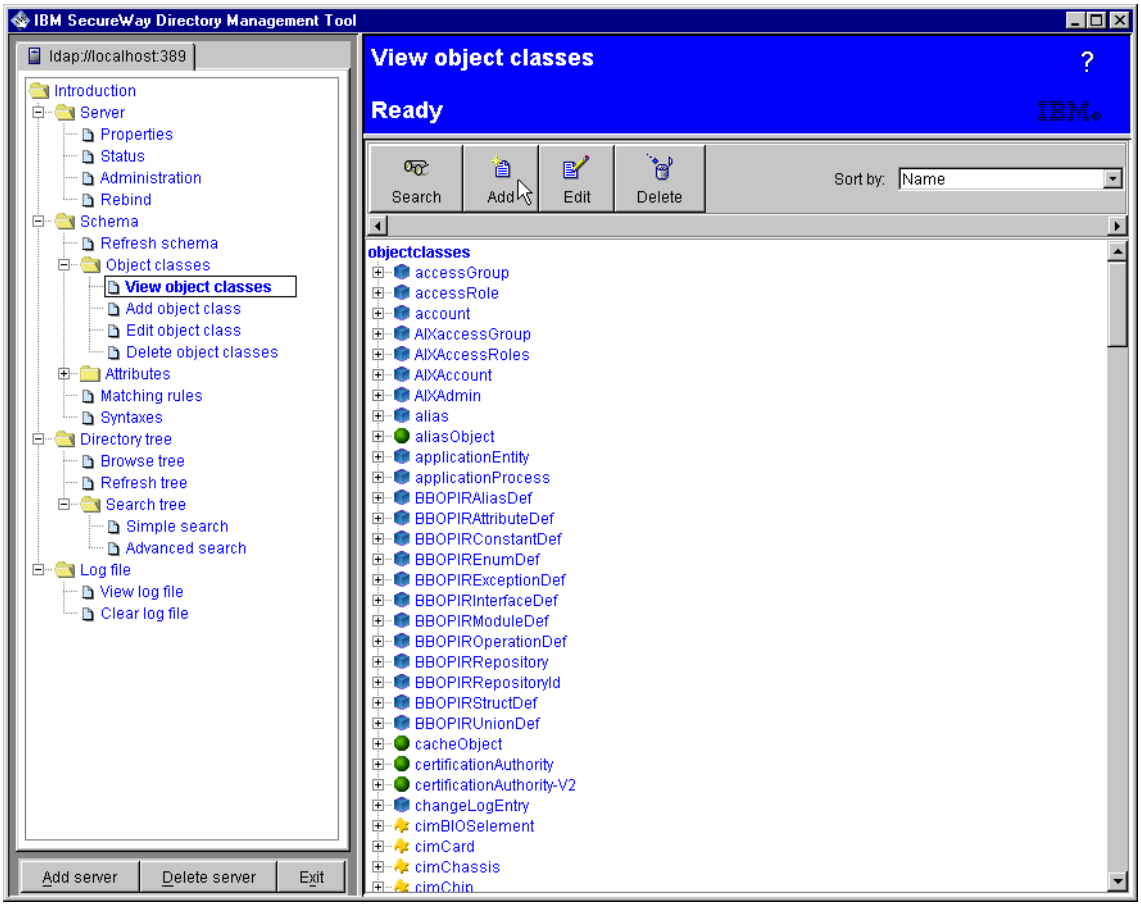

Figure 94. SecureWay - adding a class

Then we typed in the object class name, description, OID and superior object class name: see Figure 95.

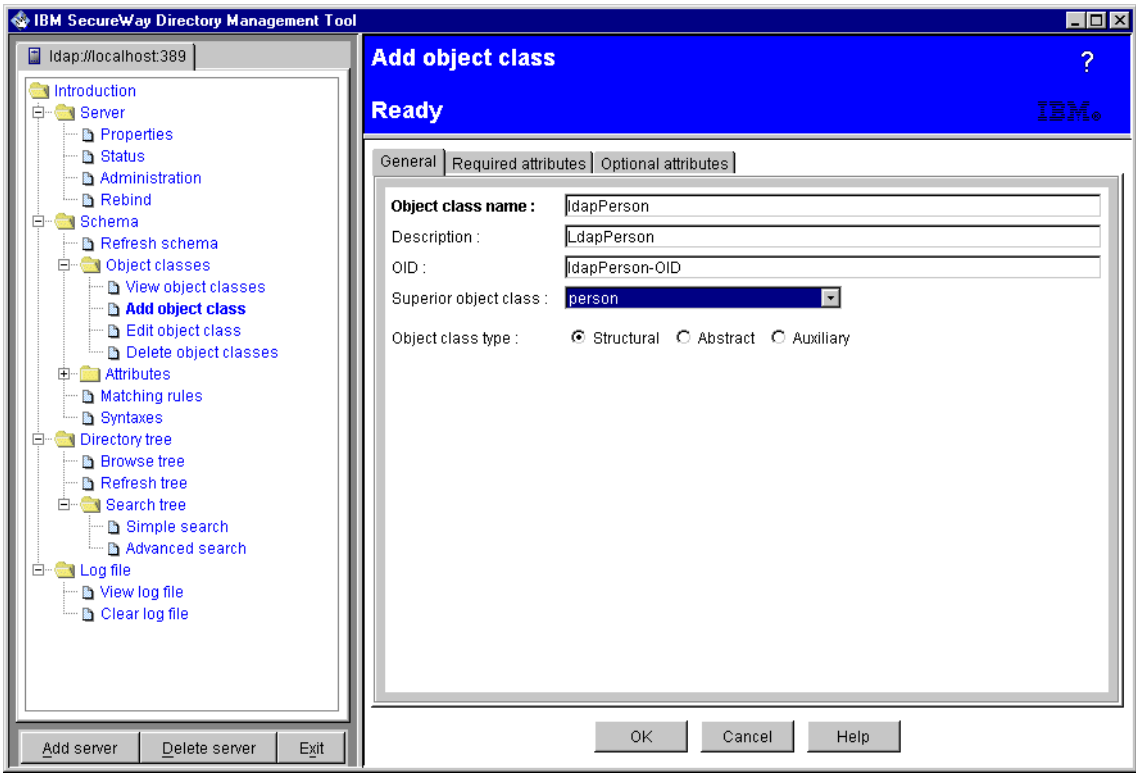

Figure 95. SecureWay - adding a class (General tab)

Then add the newly created attributes, as shown in Figure 96. Thus the IdapPerson class in both directories would have the same mandatory attributes.

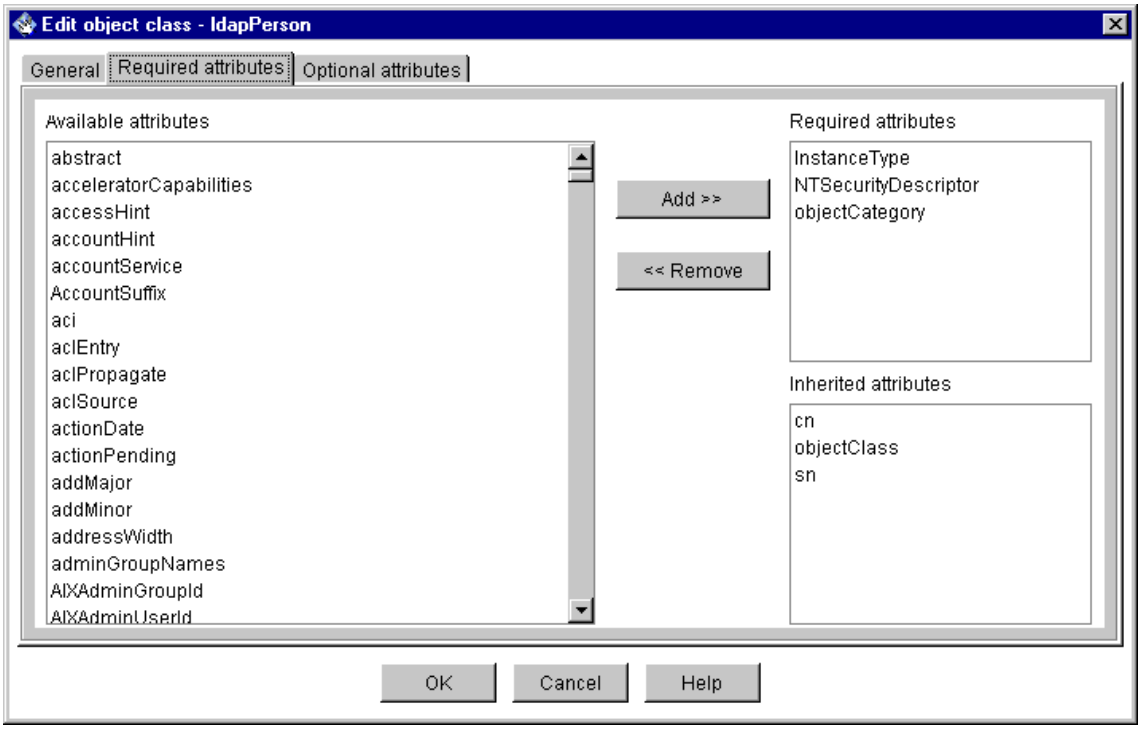

Figure 96. SecureWay - adding a class (Required attributes tab)

> **Note**
>
> In our scenario we did not add any optional attributes. But if you need to this can be done from the Optional attributes tab, as shown in Figure 96.

5.3.2 Creating new classes in Active Directory

Using the Active Directory schema console we created an ldapPerson class by right-clicking the **Classes** folder and selecting **Create Class**. See Figure 97.

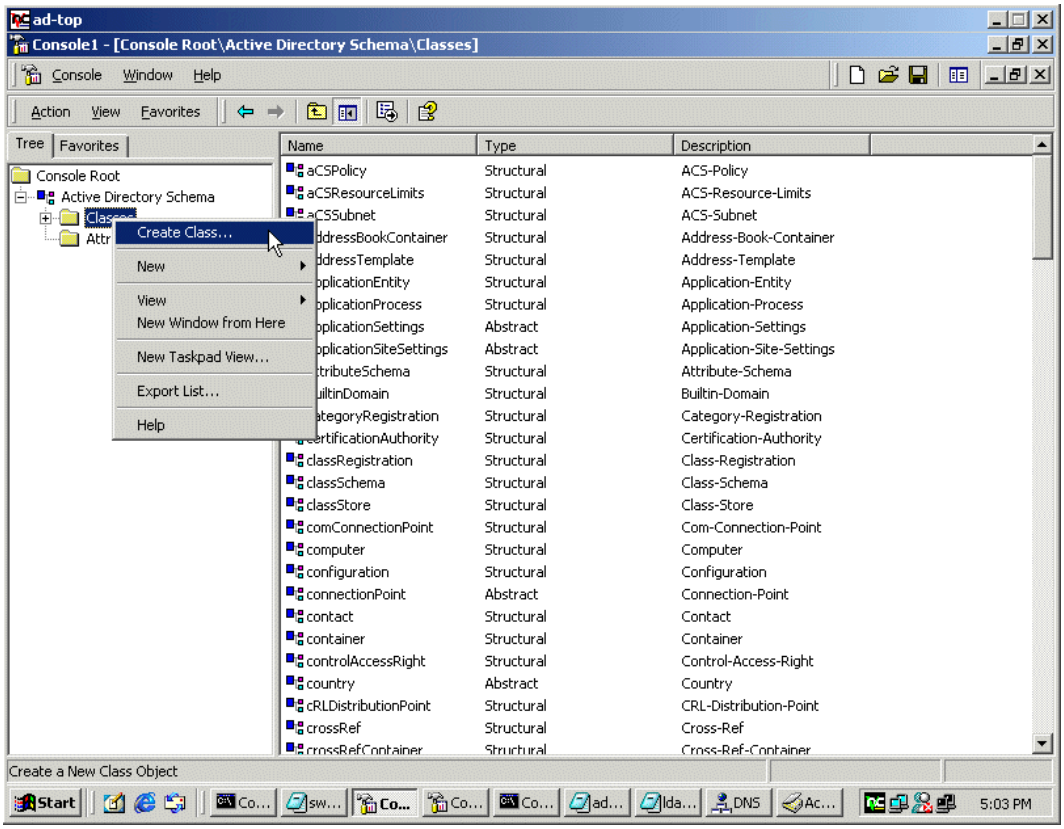

Figure 97. Active Directory - Adding a class (1)

Then we selected **Continue** to the warning; see Figure 98.

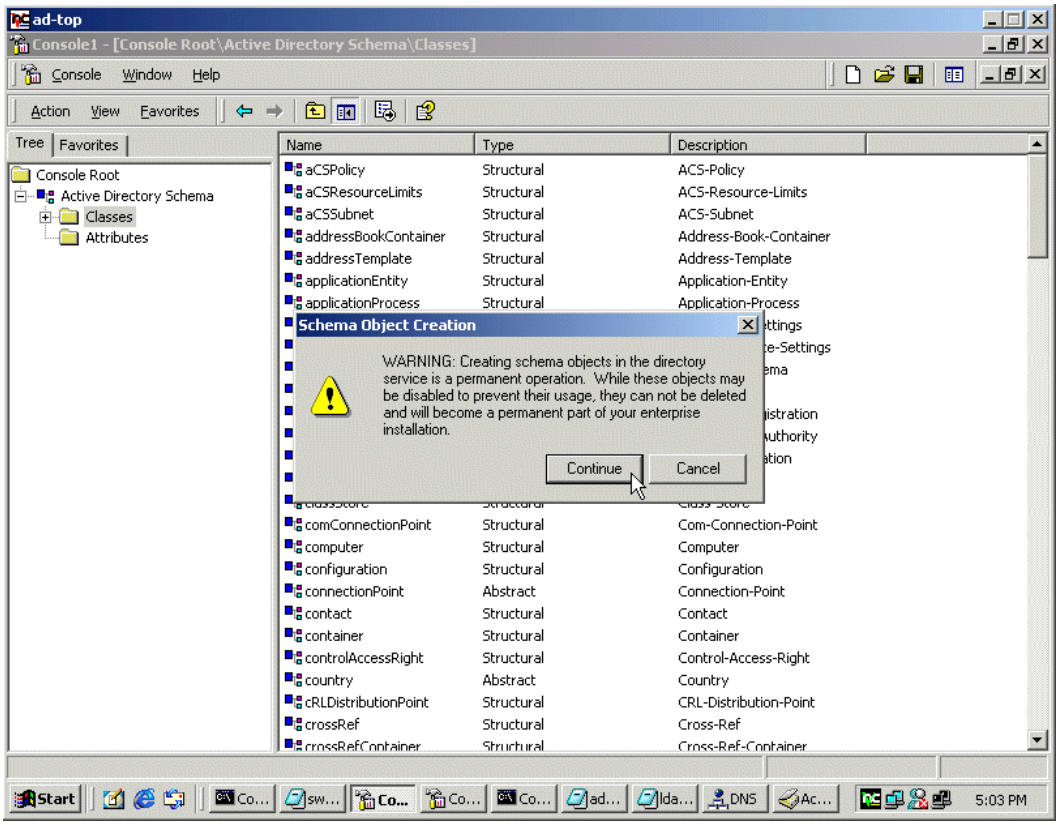

Figure 98. Active Directory - adding a class (2)

Then we typed in the Common Name, LDAP Display Name, Unique X500 Object ID, Parent Class and selected Class Type. Then we clicked the **Next** button. See Figure 99.

Note

Once a schema object is created it may be disabled, but not deleted, from the Active Directory.

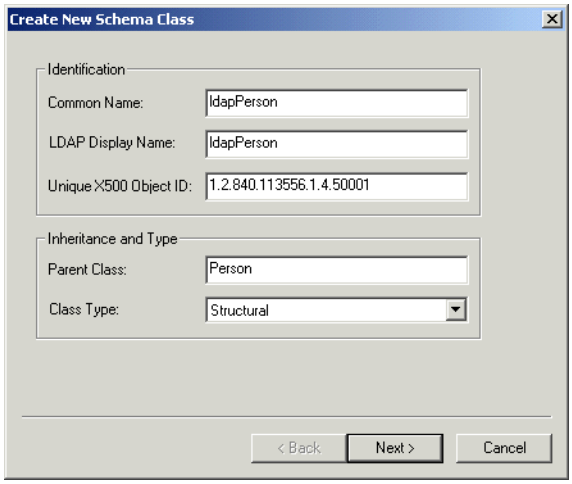

Figure 99. Active Directory - adding a class (3)

Then we selected the **Finish** button, see Figure 100.

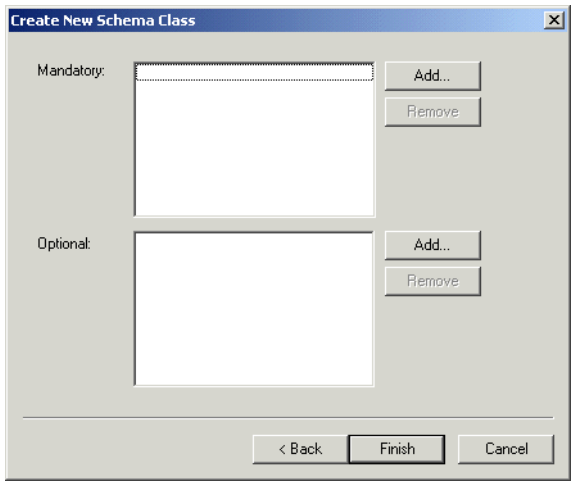

Figure 100. Active Directory - adding a class (4)

Once the ldapPerson class had been created we right clicked the ldapPerson class and selected **Object properties** and changed the category to person by selecting the **Change** button. See Figure 101 on page 130.

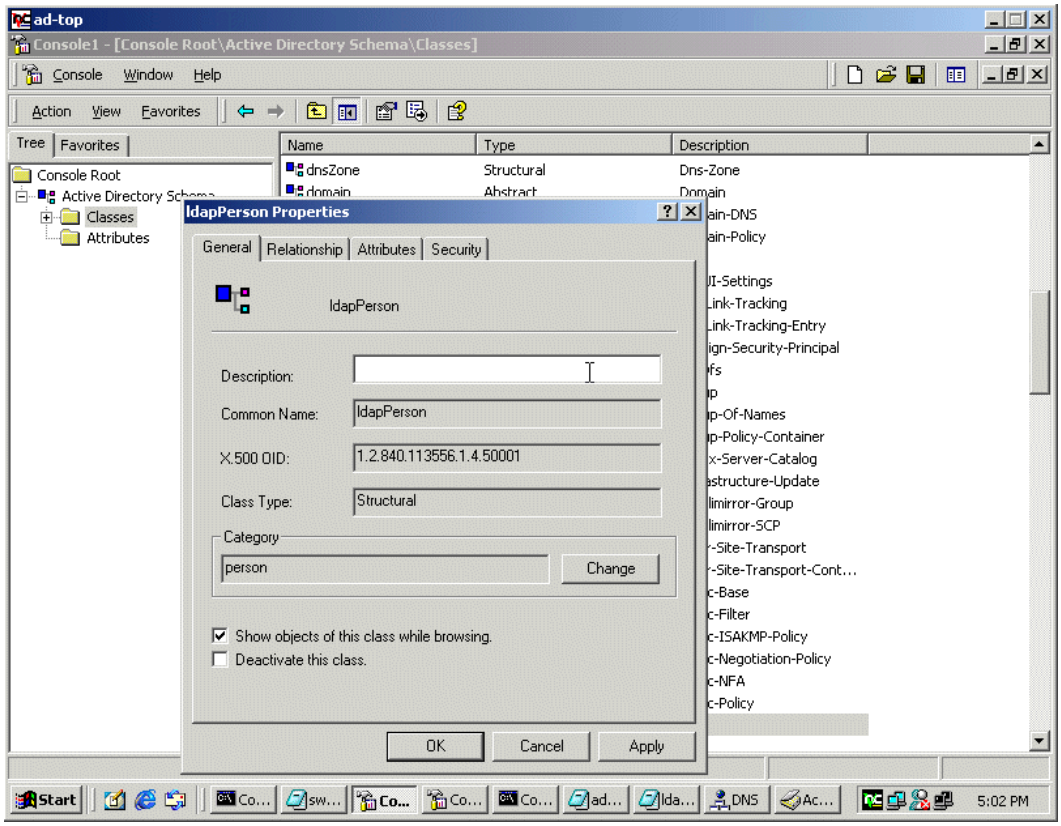

Figure 101. Active Directory - adding a class (5)

> **Note**
>
> For more infirmation look at a whitepaper *Understanding and Extending the Windows 2000 Schema* (IBM Technical Report TR 29.3210), found at
> `http://www5.pc.ibm.com/ww/me.nsf/WW-webdocs/White+Paper:+Understanding+and+Extending+the+Windows+2000+Schema`

5.4 Creating instances in SecureWay

Next we created two instances, Bernard-SW and Hani-SW, of the IdapPerson class. To add an instance in SecureWay use the IBM SecureWay Directory Management Tool and select **Browse tree** then **Add**. See Figure 102.

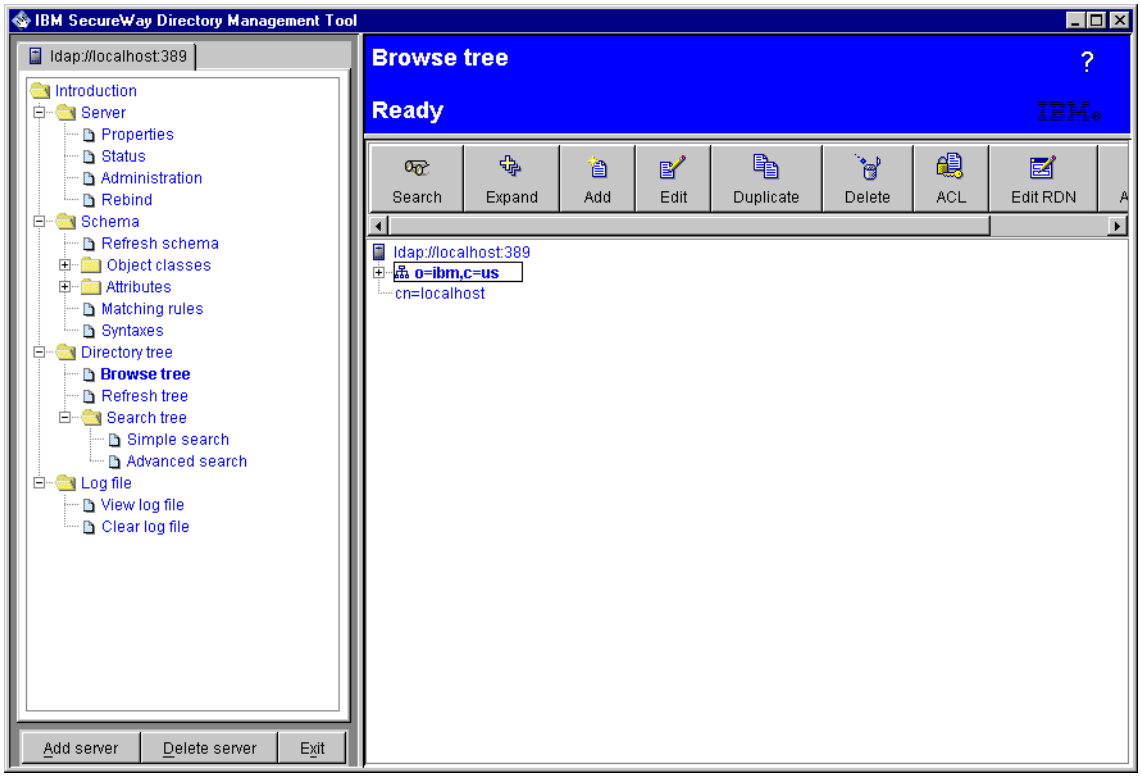

Figure 102. SecureWay - adding an instance (1)

Then type in the Parent DN and Entry RDN and select the appropriate Structural object class. See Figure 103.

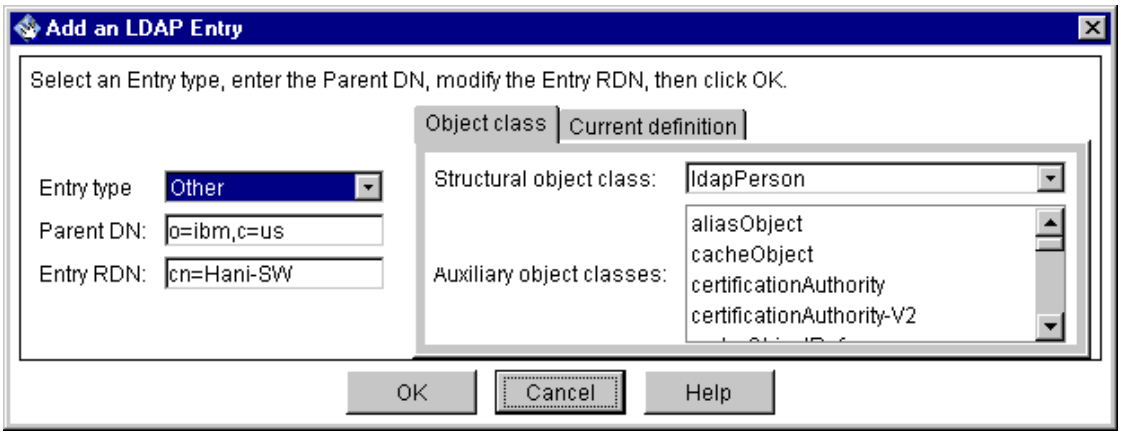

Figure 103. SecureWay - adding an instance (2)

Select the Object class and type in the DN, InstanceType, Last name, NTSecurityDescriptor and objectCategory. See Figure 104.

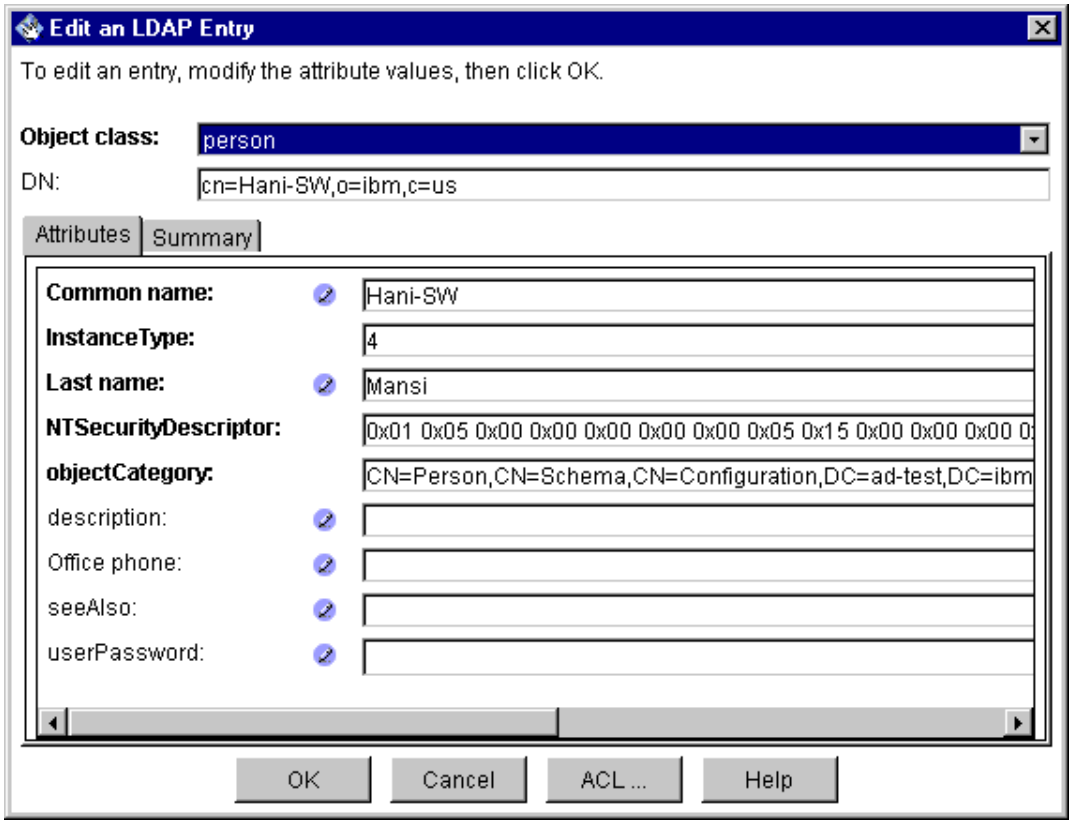

Figure 104. SecureWay - adding an instance (3)

5.5 Exporting and importing LDIF files

The following shows how to import and export files.

5.5.1 Exporting SecureWay instances

We used the following command, from a command prompt, to export the two instances to an LDIF file. For an explanation of the `ldifde` switches see Appendix J, “LDAP utilities” on page 207.

```
ldifde -f ldappeople.ldif -s ldap1 -v -r objectclass=ldapperson -c
“o=IBM,c=US” “dc=ad-test,dc=ibm,dc=com”
```

Note

The `-c` switch translates the DN suffixes for disjointed namespaces. It can be used either while exporting or while importing LDIF files. For more information on `ldifde` switches, see Appendix J, “LDAP utilities” on page 207.

5.5.2 Importing Active Directory instances

We used the following command, from a command prompt to import the LDIF file. To see an explanation of the `ldifde` switches, see J.1, “ldifde utility” on page 207.

```
ldifde -i -f ldappeople.ldif -s ad-top -v
```

5.5.3 Listing of ldappeople.ldif

```
dn: cn=Bernard-SW,cn=users,dc=ad-test,dc=ibm,dc=com
changetype: add
objectcategory:
CN=Person,CN=Schema,CN=Configuration,DC=ad-test,DC=ibm,DC=com
objectclass: ldapPerson
instancetype: 4
sn: Freund
cn: Bernard-SW

dn: cn=Hani-SW,cn=users,dc=ad-test,dc=ibm,dc=com
changetype: add
objectcategory:
CN=Person,CN=Schema,CN=Configuration,DC=ad-test,DC=ibm,DC=com
objectclass: ldapPerson
instancetype: 4
sn: Mansi
cn: Hani-SW
```

5.6 Synchronizing information

Probably, the most common operation you will want to do after the directories are populated is to keep them synchronized. We have seen in 5.5, “Exporting and importing LDIF files” on page 133 how to make the initial data load for the directories.

Now we will show how to keep them synchronized. Since developing a custom directory synchronization tool, albeit simple, would fall outside the scope of this book, we decided to use scheduled batch files (calling the IBM

ldapmodify program, which ships with the IBM SecureWay Directory) to implement this functionality.

Note

For information on how to use the ldapmodify program, as well as any other IBM ldap utilities, see Appendix J, “LDAP utilities” on page 207.

To exemplify the work of these files, we ran the commands from the Windows NT command prompt. First, we generated an LDIF export of the two instances we created (Hani-SW and Bernard-SW), using the `-c` switch of the `ldifde` command for DN translation, when appropriate (for more information on the `ldifde` command switches, see Appendix J, “LDAP utilities” on page 207).

You can see a sample file for adding a description to the Hani-SW ldapperson in SecureWay:

```
dn: cn=Hani-SW,o=ibm,c=us
changetype: modify
replace: description
description: Hani Mansi
```

Below, a sample file for doing the same to the Bernard-SW ldapperson in Active Directory:

```
dn: CN=Bernard-SW,CN=Users,DC=ad-test,DC=ibm,DC=com
changetype: modify
replace: description
description: Bernard Freund
```

Next, we imported the resulting files using the `ldapmodify` command:

```
C:\>ldapmodify -h ldap1 -p 389 -D "cn=root" -w ldap -r -v -f "ldap.ldif"
ldap_init(ldap1, 389)
replace description:
        BINARY (10 bytes) Hani Mansi
modifying entry cn=Hani-SW,o=ibm,c=us
modify complete

C:\>ldapmodify -h ad-top -p 389 -D "ad-test\administrator" -w ldap -r -v -f "ldap.ldif"
ldap_init(ad-top, 389)
replace description:
        BINARY (14 bytes) Bernard Freund
modifying entry CN=Bernard-SW,CN=Users,DC=ad-test,DC=ibm,DC=com
modify complete
```

As you can see above, the synchronization process is very simple.

5.7 What we have discovered

To exchange data between different directories there needs to be a common set of attributes and classes. Finding this common ground can be complex and time consuming but is very necessary for the integration of different directories. Although our scenario is relatively simple, it shows that it is possible to exchange data between different directories and keep the data up-to-date.

We hope that in the near future cross-vendor synchronization is standardized, making this an even simpler task.

In the meantime, schema extensibility can be achieved using manual updates or through custom applications. Throughout this scenario, we expanded on the concepts above by using an IBM-written application that generates schema import files for several directories, including the IBM SecureWay Directory, Lotus Domino and Active Directory. These schema import files contain all necessary attributes and classes for creating partial schema compatibility in such a way that applications can use the same objects for a variety of purposes.

Chapter 6. Conclusions

Throughout the writing of this book, we demonstrated several possibilities for basic directory integration, from a systems administrator point of view. We suggest that, on your environment, you start by defining clearly your goals for the directory service.

You should also keep in mind that building and maintaining a directory service should be faced not as a task, with well-defined start and finish milestones, but rather as an ongoing effort, since the directory changes and evolves along with the corporation.

Thus, we feel that a step-by-step approach (similar to the one used during this writing), where you set small goals and then you build upon them, is highly advisable. We felt that just by successfully establishing referrals, a relatively simple step (when compared to creating schema compatibility, for example), allowed us to reach a basic, but satisfactory, level of integration, so we could start reaping the benefits and then improving what we had created.

Of course, our strategy may not be adequate for every situation, but we feel that the important message is that you should carefully evaluate the business benefits of all intermediate steps toward the higher objective, which is full directory integration and compatibility.

This way, you may achieve a high level of success in your directory projects.

6.1 Challenges

Undoubtedly, the hardest challenge faced by organizations of any size trying to integrate different directory services is the schema compatibility.

While basic X.500/LDAP standard classes are a good starting point, a fair amount of work would be needed to make the various vendors' schemas compatible.

The samples below were extracted from the files that resulted from running an LDIF export (for more on the use of the `ldifde` command, see Appendix J, "LDAP utilities" on page 207) of the schema naming context in each directory (`cn=schema` for Domino and SecureWay and `cn=aggregate, cn=schema, cn=configuration,` etc. for the Active Directory). The first three show the definition for the person class, and the remaining three show the definition for the seeAlso attribute.

- IBM SecureWay Directory schema object class definition sample:

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

- Lotus Domino directory schema object class definition sample:

```
dn: cn=Schema
changetype: add
...
objectclasses: ( 2.5.6.6 NAME 'person' STRUCTURAL SUP 'Top'  MUST (
objectClass $ sn $ cn ) MAY ( userPassword $ telephoneNumber $ seeAlso $
description ) NOTES 'Person' )
...
```

- Microsoft Active Directory schema object class definition sample:

```
dn: cn=aggregate,cn=schema,cn=configuration,dc=bef-test,dc=local
changetype: add
...
objectClasses: ( 2.5.6.6 NAME 'person' SUP top ABSTRACT MUST (cn ) MAY
(sn $ telephoneNumber $ seeAlso $ userPassword ) )
...
```

- IBM SecureWay Directory schema attribute definition sample:

```
...
( 2.5.4.34 NAME 'seeAlso' SUP 2.5.4.49 USAGE userApplications )
...
```

Note

If there are some non-ASCII characters in the attributetype definitions, SecureWay returns schema attribute definitions to LDAP searches in encoded (base64) form. If this happens, we can look at its schema files (the IBM SecureWay Directory stores its schema in flat files) to find the attribute definition.

- Lotus Domino directory schema attribute definition sample:

```
...
attributetypes: ( 2.5.4.34 NAME 'seeAlso' SYNTAX
1.3.6.1.4.1.1466.115.121.1.12 NOTES 'seeAlso'))
...
```

Microsoft Active Directory schema attribute definition sample:

```
...
attributeTypes: ( 2.5.4.34 NAME 'seeAlso' SYNTAX
'1.3.6.1.4.1.1466.115.121.1.12' )
...
```

Note

IBM offers its directory in several formats. The samples above were extracted from the SecureWay Directory on Windows NT.

At the time of the writing of this book, development efforts were being made by all vendors to make the schema formats more compatible, and to port their standard schemas to each other's directory offerings.

While the samples above may seem similar at first, they have subtle differences in format that make interchangeability difficult. So, when planning for schema extensions by adding new attributes and classes, or by subclassing existing ones, the developers have to create their extensions in several different formats, for each directory that will be used by the application.

In addition, the developers also have to make sure the end result is the same on every directory. Even when they only subclass the standard X.500/LDAP classes (or, minimally, the top class, when creating brand new classes), the developers still have to be very careful with the mandatory and optional attributes. As we have seen in Chapter 5, "Scenario4: Extending schemas and synchronizing data" on page 121, different vendors add different mandatory attributes to their top class, making compatibility an even more difficult and laborious task.

6.2 The need for a standard for vendors' extensions

In addition to the abstract (or aggregate) schema format differences, there are differences in the way vendors map the attributes and classes to their physical database implementations. The next six examples, extracted from the same LDIF files as the ones in 6.1, "Challenges" on page 137, illustrate this, both for object classes and attributes.

- The IBM SecureWay Directory does not return any extended schema object class definitions to LDAP searches.
- The Lotus Domino directory does not return any extended schema object class definitions to LDAP searches.

- Microsoft Active Directory *does* return extended schema object class definitions:

```
...
extendedClassInfo: ( 2.5.6.6 NAME 'person' CLASS-GUID
'A77A96BFE60DD011A28500AA003049E2' )
...
```

- IBM SecureWay Directory extended schema attribute definition sample:

```
...
ibmattributetypes: ( 2.5.4.34 DBNAME ( 'seeAlso' 'seeAlso' )
ACCESS-CLASS NORMAL LENGTH 1000 )
...
```

- The Lotus Domino directory does not return any extended schema attribute definitions to LDAP searches.
- Microsoft Active Directory extended schema attribute definition sample:

```
...
extendedAttributeInfo: ( 2.5.4.34 NAME 'seeAlso' PROPERTY-GUID
'317A96BFE60DD011A28500AA003049E2' PROPERTY-SET-GUID
'00000000000000000000000000000000' )
...
```

As you can see, there are significant differences in schema extended class and attribute information. As a result, reaching a common schema is a very challenging task, without some kind of standardization of the schemas and schema extensions. As we mentioned in Chapter 5, “Scenario4: Extending schemas and synchronizing data” on page 121, an automated custom application for generating schema files can be an effective mechanism to establish a common ground. Those can be expensive too, so they should be an option used mostly by application vendors, as opposed to end users.

6.3 The need for directory standards

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 often store only a specific type of information, usually do not have general search capabilities, usually do not support replication and partitioning, and often do not have a full set of administration tools. An application-specific directory can be as simple as a set of editable text files, or it can 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, 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 rich 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, remote procedure call (RPC), and object request brokers (ORBs) 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 will 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. The calendar, mail, and operator notification applications can all access the same directory to retrieve an e-mail address. New uses for directory information will be realized, and a synergy will develop as more applications take advantage of the common directory.

6.4 What to do?

We recommend that you evaluate the objects and implementation needs in the early stages of development, and that you experiment a lot with scripted and manual object creation in all directories (in a controlled test environment)

before you commit to any development projects that may include extending one or more of the schemas of the directories.

Another helpful step, from a portability standpoint, is to add as much intelligence to the application itself, or to add a business logic layer that will make the underlaying directories transparent (implementing this layer, of course, is the tricky part of this strategy).

Finally, in a multi-directory, multi-vendor environment, using a metadirectory may be the safest approach, although it may also be costlier and not all metadirectory products may suit your particular needs.

Chapter 7. Future

Although a metadirectory solution might be adequate in some cases, Figure 105 illustrates some of the items that one might see on Directory Integration using LDAP in the near future.

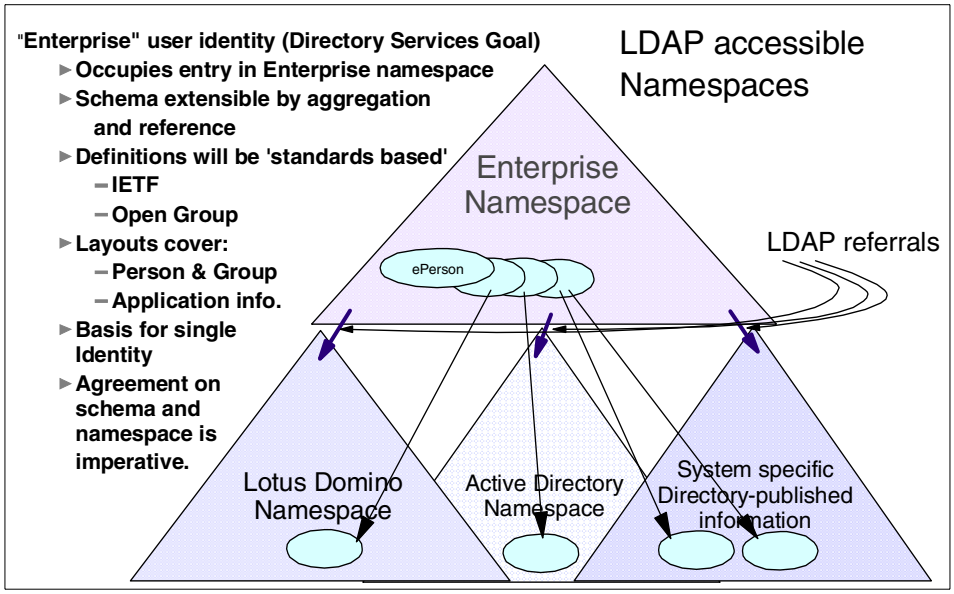

Figure 105. The enterprise namespace

An Enterprise namespace needs to cover more than just OS/390. It must cover the computing resources of the entire enterprise, including existing workgroup servers.

Many Directory Services publish and allow updates to their information via the LDAP protocol. These include:

- IBM SecureWay Directory
- Novell eDirectory (formerly known as NDS)
- Microsoft Active Directory
- Netscape iPlanet Directory Server
- Lotus Domino Directory

7.1 Converging on standards -- LDAP

Having published LDAP V2 as RFCs 1777-78 in 1995, and LDAP V3 as RFCs 2251-56 in December 1997, the two current IETF working groups dedicated to furthering the scope of LDAP-related directory functionality. The LDAP Extension (LDAPEXT) working group defines and standardizes extensions to the LDAP V3 protocol and extensions to the use of LDAP on the Internet. The planned extensions include the following areas:

- Access controls
- Server-side sorting of search results/paged retrieval of search results
- Language tags
- Dynamic directories
- Referral and knowledge reference maintenance
- LDAP server discovery
- LDAP APIs
- CLDAP
- Signed directory information

Other areas such as deployment and schema definition and review will be handled by other groups if and when they turn out to be necessary for the deployment of LDAP and feasible for the group to tackle.

The second group, LDUP, addresses the issues of replication of data across servers running different implementations, which is becoming an important part of providing a distributed directory service as LDAP V3 becomes more widely deployed. The LDAP V3 community to date has focused on standardizing the client/server access protocol, and so the LDUP group is chartered to standardize LDAP V3-based replication according to the following two models:

- **Multi-master replication:** entries can be written and updated on any of several replica copies, without requiring communication with other masters before the write or update is performed.
- **Master-slave, or single-master replication:** assumes only one server (the single-master) is allowed write-access to replicated data. (Note that master-slave replication can be considered a proper subset of multi-master replication.)

The new replication architecture will support both these approaches.

The LDUP working group first developed a set of requirements for LDAP V3 directory replication, and then wrote an applicability statement defining scenarios on which replication requirements are based. Six areas of working group focus have been identified, each leading to one or more documents to be published:

- LDAP V3 Replication Architecture
- LDAP V3 Replication Information Model
- LDAP V3 Replication Information Transport Protocol
- LDAP V3 Mandatory Replica Management
- LDAP V3 Update Reconciliation Procedures
- LDAP V3 Profiles

It is anticipated that consensus on the LDUP work will be reached during 2001, with the anticipated uptake by vendors over the following two years.

7.1.1 Common data definition -- DSML

Directories typically store and manage information about each user in an enterprise - including names, addresses, phone numbers and access rights. Directories are increasingly storing metadata about available Web services, what they do, what they require for input, how to execute them, what the results will be, who wrote them and how to pay for them. Combined with the power of eXtensible Markup Language (XML), the Internet's common language for e-business - this information enables whole new classes of individually tailored applications for e-commerce. The definition of the XML schema for describing directory structure and data is Directory Services Markup Language (DSML).

Applications consume DSML documents as they would XML because DSML is a subset of XML. Applications can transmit DSML documents to other DSML-enabled applications on the Internet. This process effectively extends LDAP across firewalls to any Internet transport protocol, such as HTTP, FTP, or SMTP, which is a major benefit for business-to-business (B2B) efforts.

DSML is also a major step forward in facilitating interoperability between different vendors' products, by describing their contents in XML. DSML-compliant directories can publish schema information as an XML document which can then be shared by other directories or applications. For example, account information can be maintained across multiple business partners, regardless of the underlying directory structure on each partner's site.

By leveraging the XML/DSML standards, applications can be enabled to react quickly to the needs of business, while leveraging a solid foundation of interoperability with back-end systems. The combination of LDAP and XML provides data in a way that allows easy integration within both new and existing applications; while LDAP provides a means for accessing directory information, DSML provides the means for reading and understanding directory content in XML. So, DSML provides a standard for creating XML documents from the information that LDAP delivers.

A group of commercial vendors, comprising AOL (represented by Netscape), Bowstreet, IBM, Microsoft, Novell, Oracle and Sun came together in July 1999 to announce the DSML Forum, and following the release of DSML 1.0 have approached the Organization for the Advancement of the Structured Information Standards (OASIS) (see `http://www.oasis.org`), to charter a Technical Committee to enhance and refine the DSML standard, and to continue development of DSML in an open forum. All of the original DSML founding member organizations have already joined the process, as well as numerous DSML supporting members and new interested parties, including Lotus.

7.2 Leveraging directory services -- Directory-Enabled Networks

The Directory Enabled Network (DEN) specification is designed to provide the building blocks for more intelligent networks by mapping users to services, and mapping business criteria to the delivery or network services. This will enable applications and services to transparently leverage network infrastructure on behalf of the user, empower end-to-end services, and support distributed network-wide service creation, provisioning and management.

DEN defines a directory as a centralized repository that defines the relationship of users and applications to network services. DEN builds intelligent networks and networked applications that are managed holistically by associating users and applications to network services and according to a consistent and rational set of policies. This will result in a generation of cross-domain network applications and services that are intelligent and self-managing.

DEN is part of the Distributed Management Task Force's (DMTF) (see `http://www.dmtf.org`) Common Information Model (CIM) standard to model functionality and management of network elements. DEN enables a company to manage its network as a single system and provides interoperability, data sharing, and transparency of the data source for cross-domain solutions. This

is the main point of DEN: managing the network as a whole, from the hosts to the physical components. The ultimate goal of DEN is a self-managing network; it behaves according to a defined set of rules (policies) to ensure less deployment and maintenance costs.

We can group the DEN effort's focus in three main areas: the physical infrastructure, network services, and hosts. Table 6 briefly shows the scope of each area:

Table 6. DEN main areas

Area	Scope
Physical Infrastructure	Inventory and asset tracking
	ACL management (mainly, router ACLs)
	Performance and fault management
Network Services	Quality of service (QoS)
	Policy-based routing
	Dial-in remote access and VPNs
	IP address management
Hosts	Desktop and server management

The initial LDAP mappings for the CIM Core Schema as well as the CIM Schema V2.3, which now includes the User, Support, and Diagnostic models that were completed in March 2000. The release of the LDAP mapping for the CIM Core and Physical Schema, as well as the CIM to LDAP mapping guidelines document, were completed in June 2000. Additional mapping specifications (DEN 4.0, CIM 2.5) are scheduled to be released by November 2000. Those mappings should include LDAP mappings for these elements: core, physical, network, system, device, core policy, QoS policy, and IPSec. The completion of this work should open the way for the vendor community to provide DEN-based products.

What is missing?

Two missing capabilities from the current LDAP standards that are seen as a hindrance to the DEN objective of implementing policy-based network architectures are change notification and transactional integrity. These are currently under study by the IETF, and should be implemented in the next version of the LDAP standards, as discussed in 1.1, "What is a directory?" on page 2 and 7.5, "Directories and databases" on page 152.

7.3 Directory-enabled applications

A directory-enabled application is an application that uses a directory service to improve its functionality, ease of use, and administration. Today, many applications make use of information that can be stored in a directory. For example, consider a group calendar application that is used to schedule meetings of company personnel in different conference rooms.

In the worst case, the calendar application does not use a directory service at all. If this were the case, a user trying to schedule a meeting would have to remember the room number of every conference room that might be appropriate for the meeting. Is the room big enough, does it have the necessary audio and video equipment, and so on? The user would also have to remember the names and e-mail addresses of every attendee that needs to receive a meeting notice. Such an application would obviously be cumbersome to use.

If conference room information (size, location, special equipment, and so on) and personnel information (name, e-mail address, phone number, and so on) can be accessed from a directory service, the application will be much easier to use. Also, the functionality of the application can be improved. For example, a list of all available conference rooms meeting the size and equipment requirements can be presented to the user. Table 7 shows some types of applications that can benefit from being directory-enabled and the benefits of directory-enabling them:

Table 7. Benefits of directory-enabling applications

Type of application	Benefits of directory-enabling
Line-of-business	Enterprise policies
	Separates infrastructure from business logic
	Locate components (objects) dynamically
	Quality of service (QoS)
	Roaming users
	Dynamic service publication and location

Type of application	Benefits of directory-enabling
Document management	Dynamic workflow server publication and location
	Separates infrastructure from business logic
	The directory contains most of the needed infrastructure
	Leverages policies for business authorization
Middleware (message queuing, distributed transaction processing)	Object location (brokerage)
	Dynamic server location
	Dynamic object deployment
	Automatic fault tolerance
Traditional client-server applications	The directory contains most of the needed security infrastructure
	Dynamic server location
Infrastructure products (storage management, systems management)	Enterprise policies
	Dynamic server and collection point location

The IBM SecureWay Directory and Client SDK (described in Chapter 1.5, “The IBM SecureWay Directory and Client SDK” on page 32) supports two ways of directory-enabling applications: JNDI and the C API.

The Java Naming and Directory Interface (JNDI) was defined by Sun Microsystems, Inc. It 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 C LDAP API is, as the name implies, an API library for C language applications. As the JNDI, it can be used for establishing connections, performing searches, and parsing the results.

Microsoft offers its own API, called Active Directory Services Interface (ADSI).

Today, several leading ERP companies, such as SAP, Baan and J. D. Edwards, have (or are in the process of) directory-enabled their applications, to bring the focus of their development efforts to the business solutions,

instead of the underlying infrastructure. This is a trend that should, in the short term, spread to other vendors' as well as customers' applications.

7.4 Metadirectories: Using LDAP for authentication and single sign-on

"Why is a directory service important?" on page 2 discusses the issues that lead to creating several directories. As the number of Web servers, application servers, access servers and so on in an intranet grows explosively, a concern arises: if you manage user IDs and passwords individually, users will have many combinations of user IDs and passwords, as shown in Figure 106.

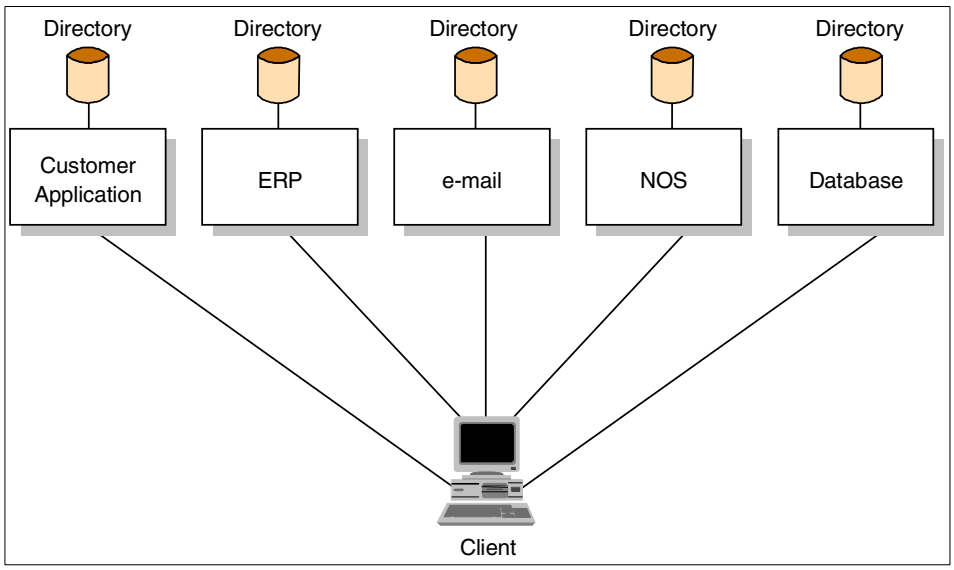

Figure 106. Several disjointed directories

Consequently, the users tend to set passwords that are easy to remember, for example, a girlfriend's name, favorite singer's name, birthday and so on. This undoubtedly makes your systems or applications vulnerable.

For this reason, LDAP directories and metadirectories are one of the most effective solutions to this problem (other important almost-metadirectory applications include "hire-and-fire" solutions, directory synchronization solutions, global address book solutions, and some e-commerce applications). The reason is that you can centralize the location of user IDs and passwords for systems or applications by using LDAP. Some Web servers and application servers already have the ability to access LDAP

directories for authentication. Therefore, even if you have a large number of Web servers and application servers, users can access every server using only one set of user IDs and passwords extracted from the LDAP directory. This makes it easier for users to create a strong password that is harder to guess, because they need to remember only one, as shown in Figure 107.

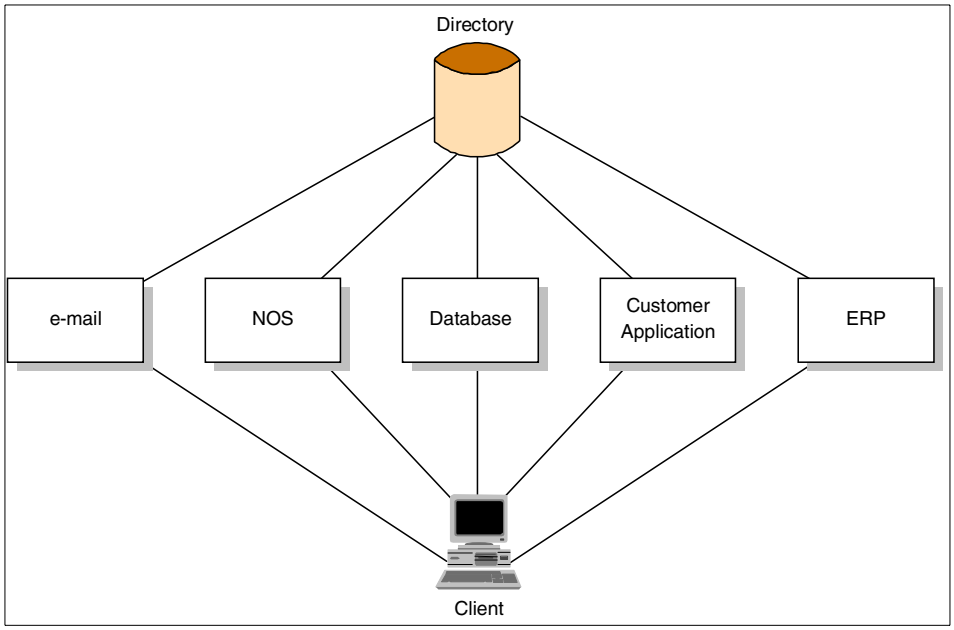

Figure 107. An integrated enterprise directory

Because user IDs and passwords are very critical information, it is desirable that this sensitive information be encrypted. LDAP provides a number of secure authentication mechanisms including Secure Sockets Layer support. Therefore, you can encrypt user IDs and passwords stored in the LDAP directory through the SSL communication channel between the LDAP clients and LDAP servers and consequently prevent hackers from guessing passwords. The IBM HTTP Server for OS/390, for example, can retrieve user IDs and passwords from the LDAP directory.

This trend will eventually lead to the complete elimination of user IDs and passwords. By using certificates instead of ID/password pairs, companies can better secure their networks, because the use of certificates makes brute-force attacks much harder (and thus, expensive). There are no easy-to-guess passwords, no social engineering (guessing passwords based on personal information), and no dictionary attacks. These certificates can even be stored on smart cards for added physical security and better mobility.

Several companies implement these solutions today, at least for network sign-on. Coupled with the growth of directory-enabled applications, we expect ever-increasing support for these solutions.

7.5 Directories and databases

In “What is a directory?” on page 2, we explored many of the differences and similarities between directories and databases. These may lead to the suspicion that a directory is no more than a limited-function database. This is indeed partly true, since one of the important design goals of a directory service is that it can be accessed and used from relatively small and simple applications. In fact, certain vendor products, such as IBM’s SecureWay Directory, use a relational database under the cover to implement the functions. LDAP as a standard does not specify the underlaying database. Each vendor chooses their implementation based on their strengths. IBM being in the large server market and addressing scalability requirements has chosen a relational database as the underlaying database.

Proposals are being discussed in the standards bodies to add some functions to future versions of LDAP that currently are specific to databases, such as support for transactional updates.

Depending on the future growth and development of these technologies (directories and databases), we may see them merging. This, in turn, would mean the definitive incorporation of directories as simple database applications, tuned for read, instead of write, performance.

An enterprise directory service will often be multiple directory servers each responsible for a portion of the directory objects and entries. It is highly unlikely that an organization will have one single software directory server to provide its enterprise directory service.

Appendix A. The RootDSE

The RootDSE is a standard attribute defined in the LDAP 3.0 specification. It contains information about the directory server, including its capabilities and configuration.

Below is an extract of section 3.4, “Server-specific Data Requirements” of the LDAP 3.0 RFC (RFC2251):

“An LDAP server MUST provide information about itself and other information that is specific to each server. This is represented as a group of attributes located in the root DSE (DSA-Specific Entry), which is named with the zero-length LDAPDN. These attributes are retrievable if a client performs a base object search of the root with filter "(objectClass=)", however they are subject to access control restrictions. The root DSE MUST NOT be included if the client performs a subtree search starting from the root.*

Servers may allow clients to modify these attributes.

The following attributes of the root DSE are defined in section 5 of RFC2252. Additional attributes may be defined in other documents.

namingContexts: naming contexts held in the server. Naming contexts are defined in section 17 of X.501.

subschemaSubentry: subschema entries (or subentries) known by this server.

altServer: alternative servers in case this one is later unavailable.

supportedExtension: list of supported extended operations.

supportedControl: list of supported controls.

supportedSASLMechanisms: list of supported SASL security features.

supportedLDAPVersion: LDAP versions implemented by the server.

If the server does not master entries and does not know the locations of schema information, the subschemaSubentry attribute is not present in the root DSE. If the server masters directory entries under one or more schema rules, there may be any number of values of the subschemaSubentry attribute in the root DSE.”

Directory vendors add other informational records as well, as they see fit.

Appendix B. Client tools

This appendix contains supplemental installation instructions for the Windows 2000 tools used throughout this book. The tools are necessary to reproduce the steps performed in all scenarios covered in this publication.

B.1 Installing Windows 2000 support tools

Two of the tools most used in our scenarios were the ADSI Edit MMC Snap-In, used for making changes to the Active Directory, and the LDP utility, a simple LDAP client that was used extensively for testing and to simulate client activity.

Follow these steps to install these tools:

1. Insert the Windows 2000 CD-ROM.
2. Click the **Start** button and then select **Run**.
3. Type the path to the installation wizard, replacing F: with your CD or DVD drive letter.

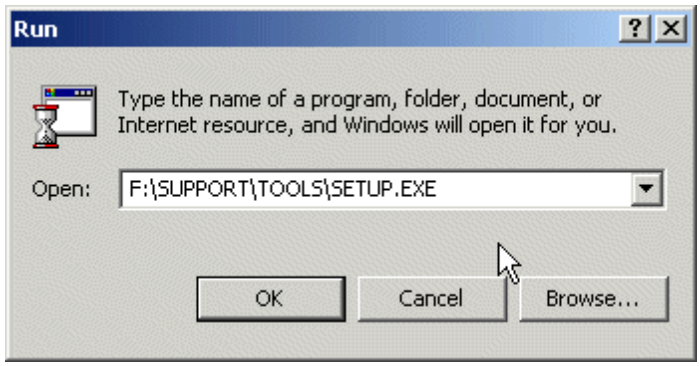

Figure 108. Starting the Windows 2000 support tools setup wizard

4. You should see the wizard's initial window:

Figure 109. The Windows 2000 support tools setup wizard

5. You should use all the defaults for the following windows.
6. At the end, you should see this window:

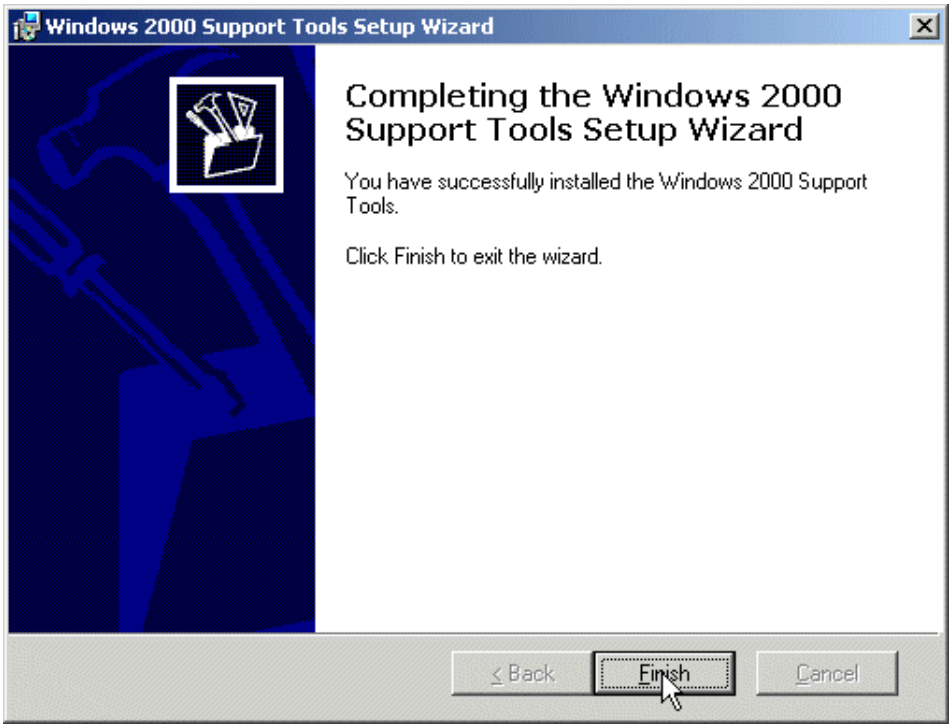

Figure 110. Completing the Windows 2000 support tools setup wizard

B.2 Installing Windows 2000 administrative tools

To install the Windows 2000 administrative tools, including the Active Directory Schema MMC snap-in, click the **Start** button and then select **Run**.Then type the path to the installation wizard, replacing **C:** with the appropriate drive letter.

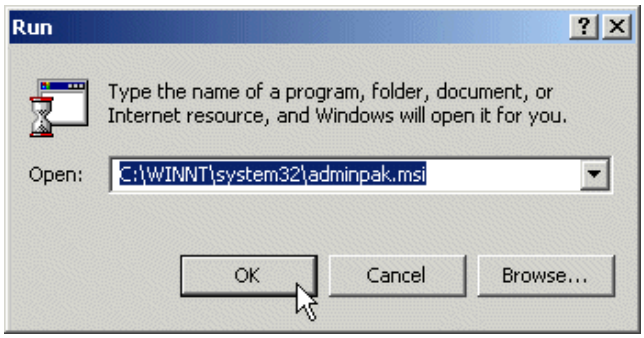

Figure 111. Starting the Windows 2000 administrative tools installation

The installation wizard will start, as shown in Figure 112.

Figure 112. The Windows 2000 administration tools setup wizard

Click **Next**, and the setup options window appears (Figure 113).

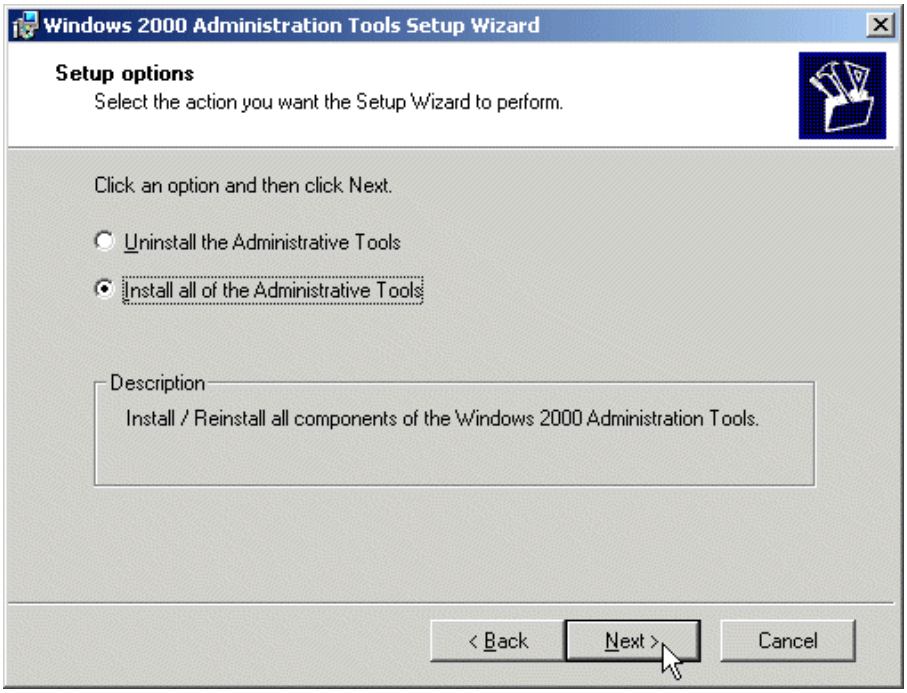

Figure 113. The setup options window

Select the **Install all of the Administrative Tools** option, and click **Next**. The administrative tools will be installed and the wizard will finish (Figure 114).

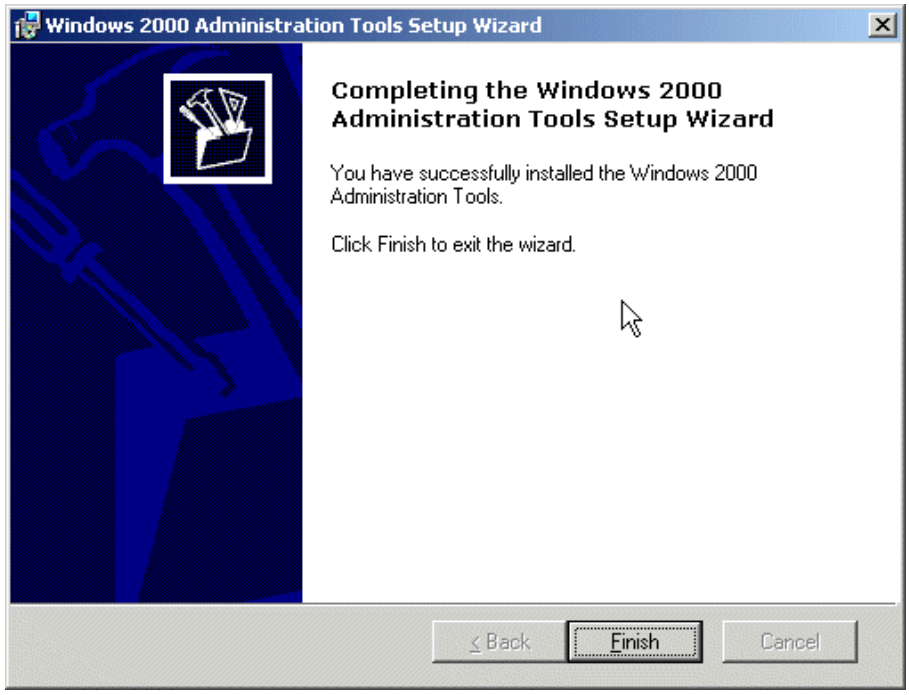

Figure 114. Finishing the administrative tools installation

Appendix C. Setting up MMC consoles

This appendix contains instructions for creating MMC consoles containing the most important management snap-ins for this book's purposes. These instructions can also be used to add those snap-ins to your own custom MMC consoles.

C.1 Setting up the ADSI Edit snap-in

To set up the ADSI Edit snap-in, open the MMC utility:

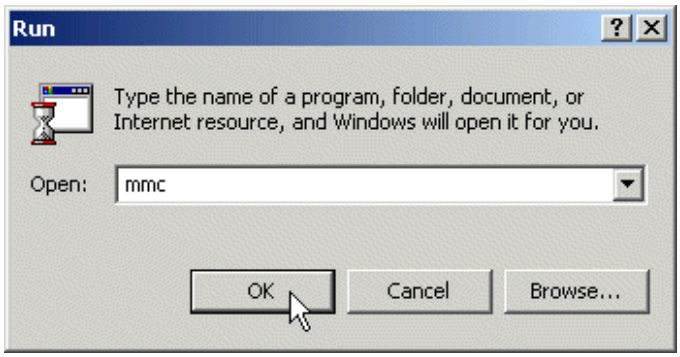

Figure 115. Starting MMC

Then select **Console**, **Add/Remove Snap-in**.

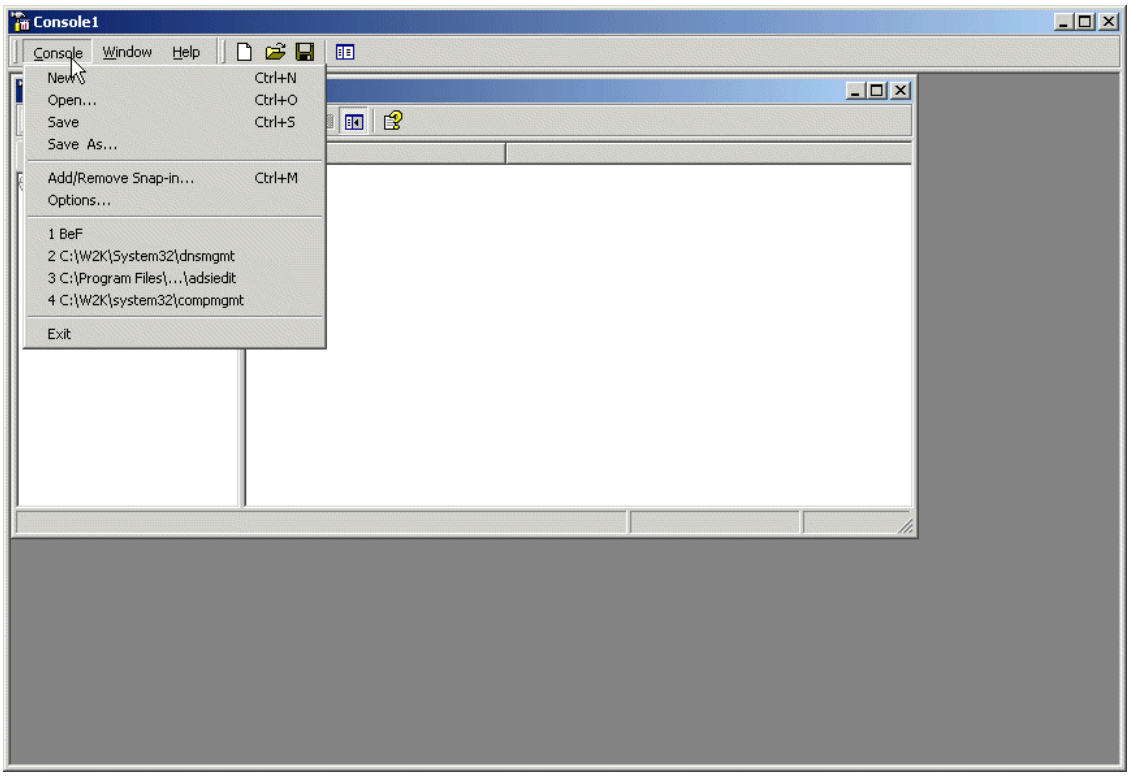

Figure 116. An empty MMC console

At the Add/Remove Snap-in dialog box shown in Figure 117, choose the **Add** button.

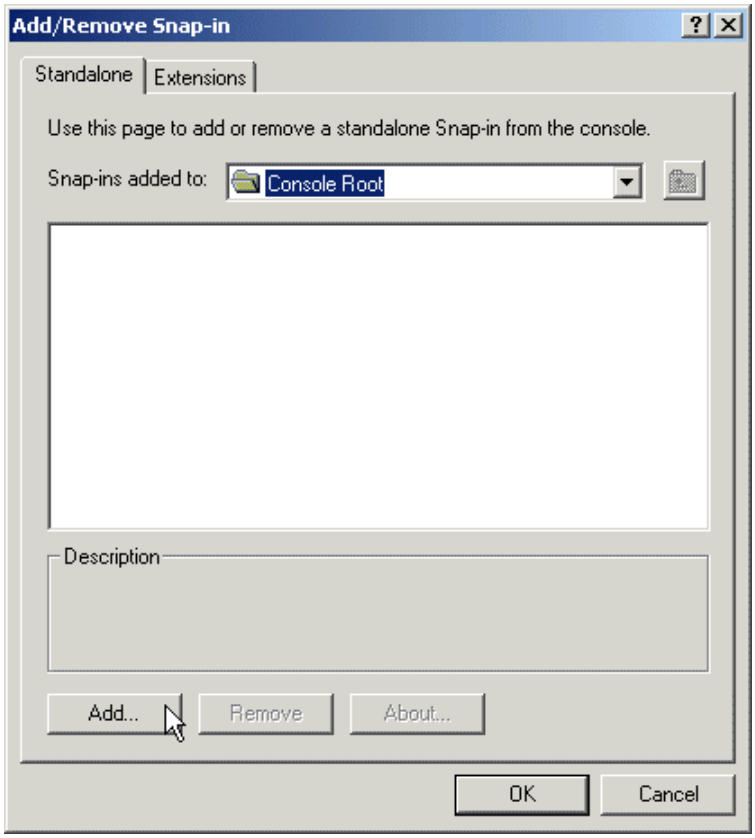

Figure 117. The Add/Remove Snap-in dialog box

In the windown shown in Figure 118, select **ADSI Edit** from the list of available snap-ins and click **Add** and **Close** to close the Add Standalone Snap-in dialog box.

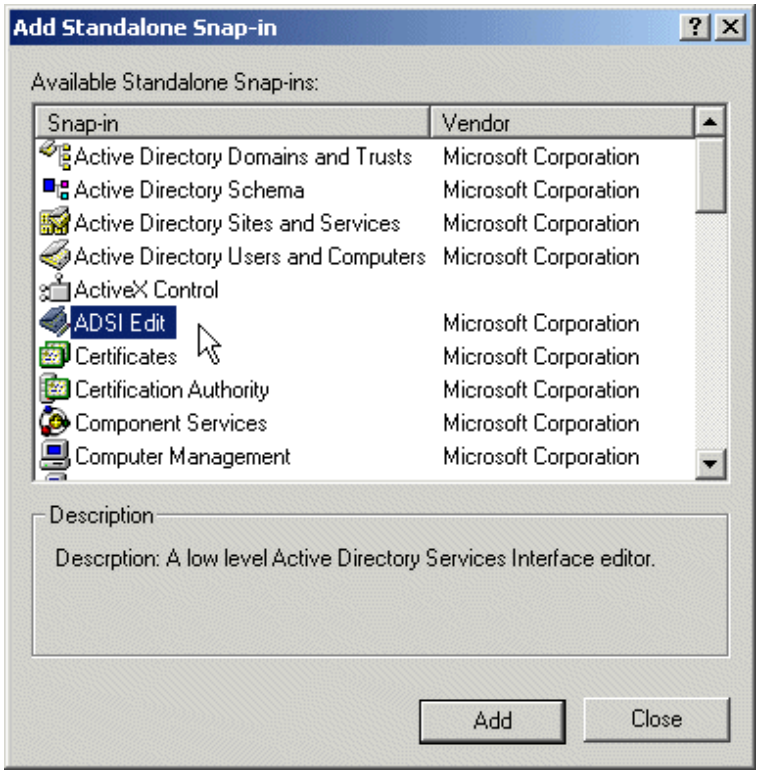

Figure 118. Adding ADSI Edit

At this point, you may want to choose to add other snap-ins as well for your console, or you may click **OK** to close the Add/Remove Snap-in dialog box.

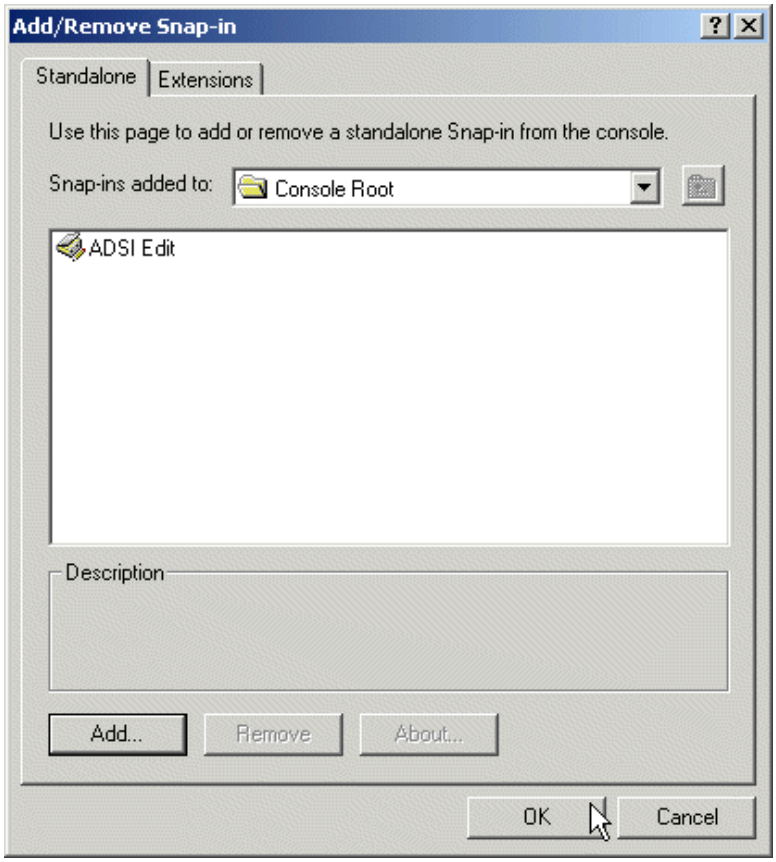

Figure 119. Closing Add/Remove Snap-in

Your window should now look like the one in Figure 120.

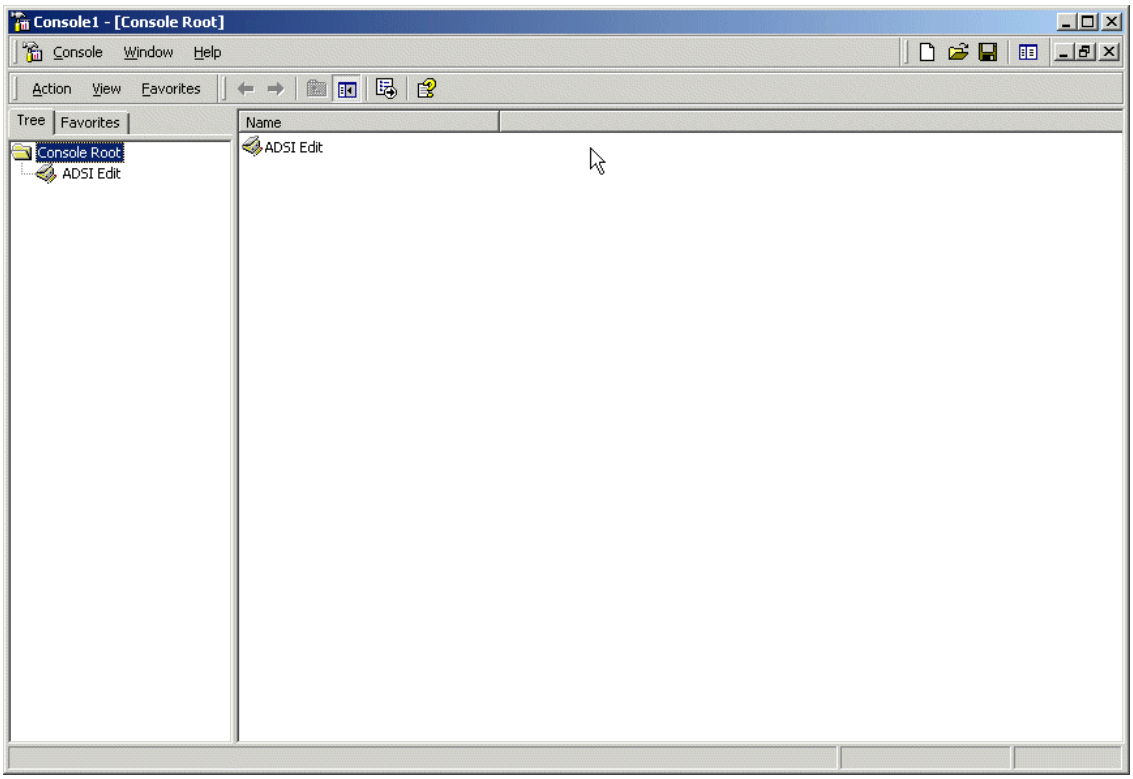

Figure 120. An ADSI Edit MMC Console

C.2 Setting up the Active Directory Schema snap-in

To set up the Active Directory Schema snap-in, open the MMC utility:

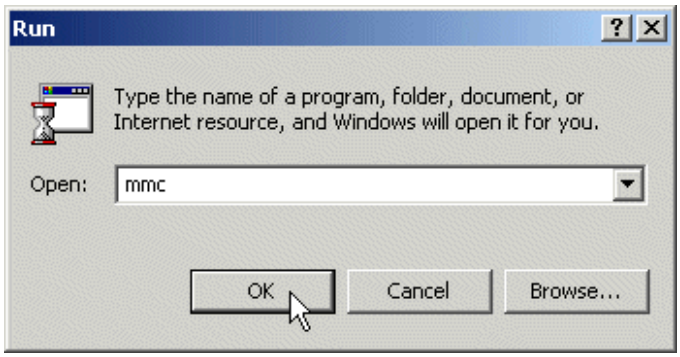

Figure 121. Starting MMC

Then, select **Console**, **Add/Remove Snap-in**. See Figure 122.

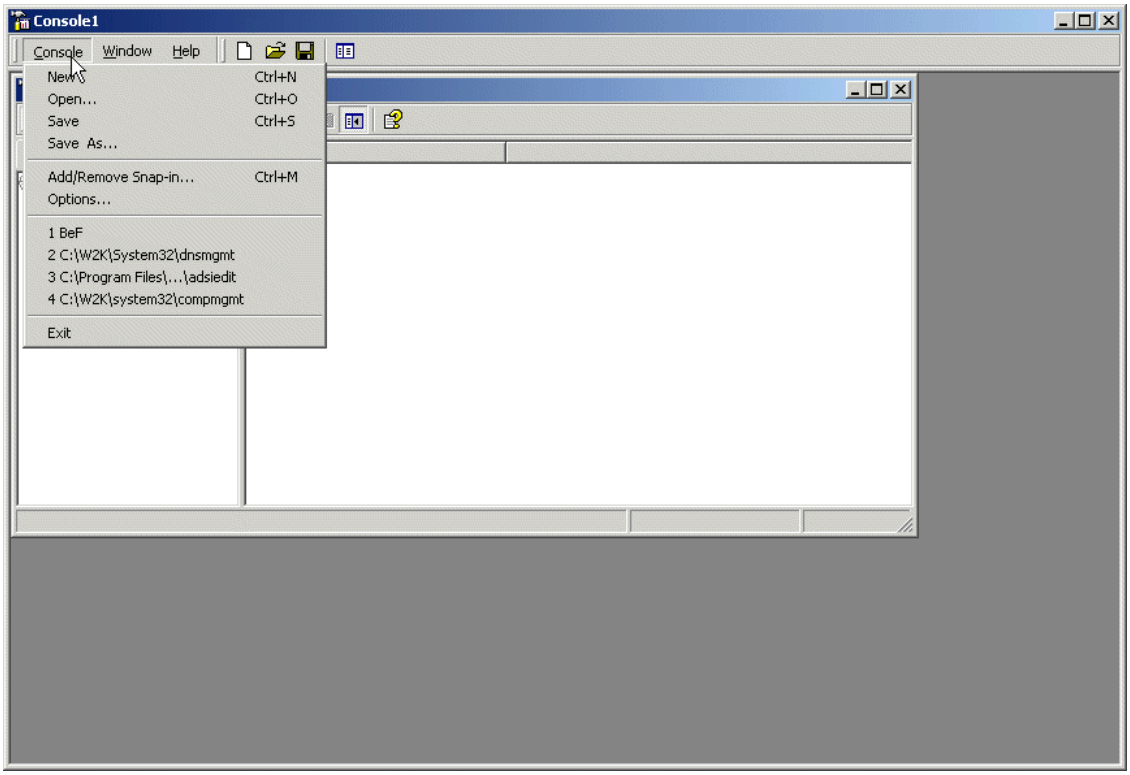

Figure 122. An empty MMC console

At the Add/Remove Snap-in dialog box shown in Figure 123, choose the **Add** button.

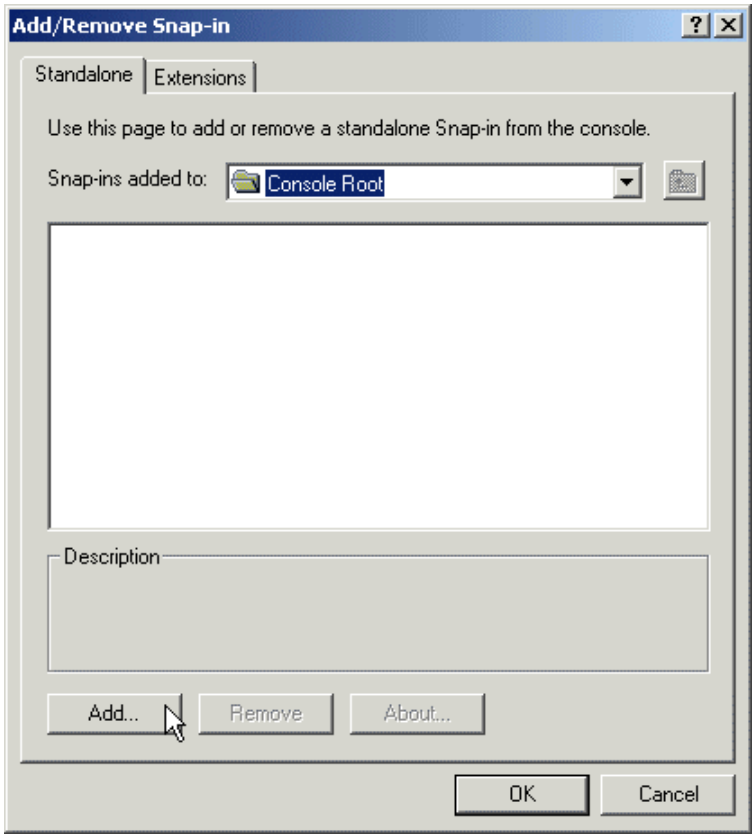

Figure 123. The Add/Remove Snap-in dialog box

As shown in Figure 124, select **Active Directory Schema** from the list of available snap-ins and click **Add** and **Close** to close the Add Standalone Snap-in dialog box.

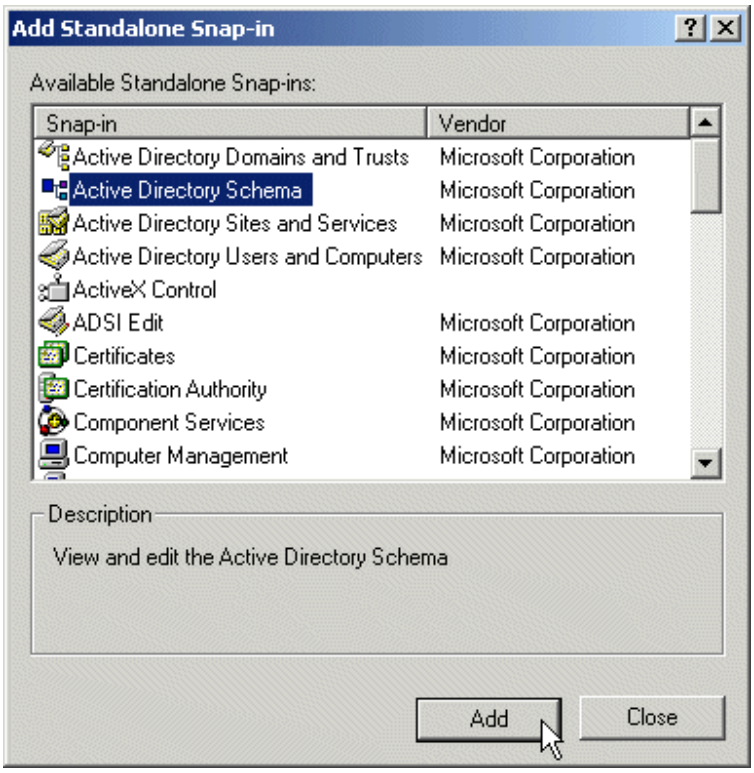

Figure 124. Adding the Active Directory Schema snap-in

At this point, you may want to choose to add other snap-ins as well for your console, or you may click **OK** to close the Add/Remove Snap-in dialog box.

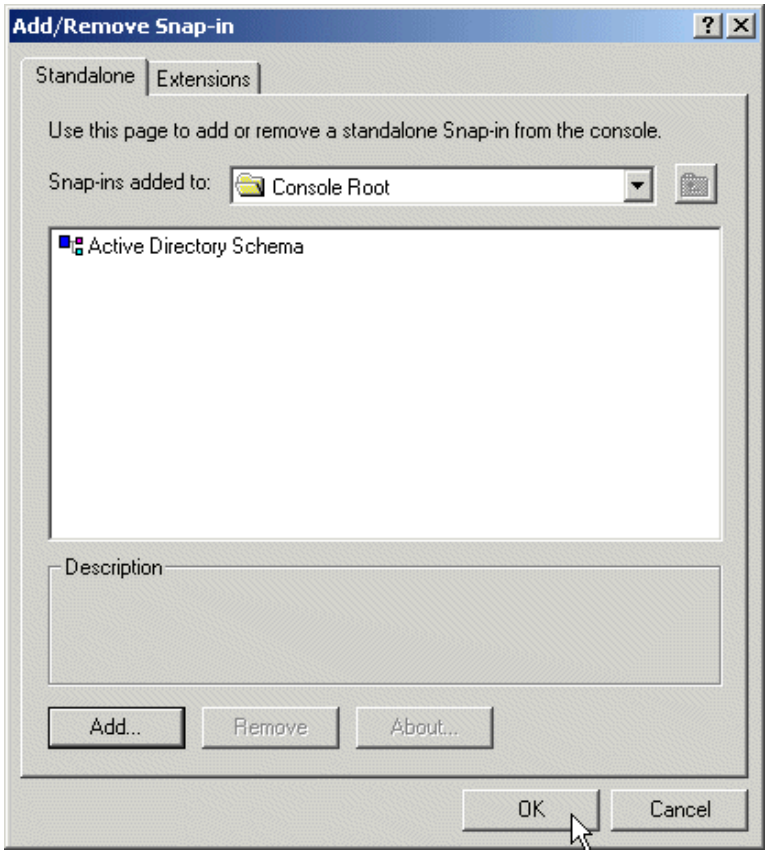

Figure 125. Closing Add/Remove Snap-in window

Your window should now look like the one in Figure 120.

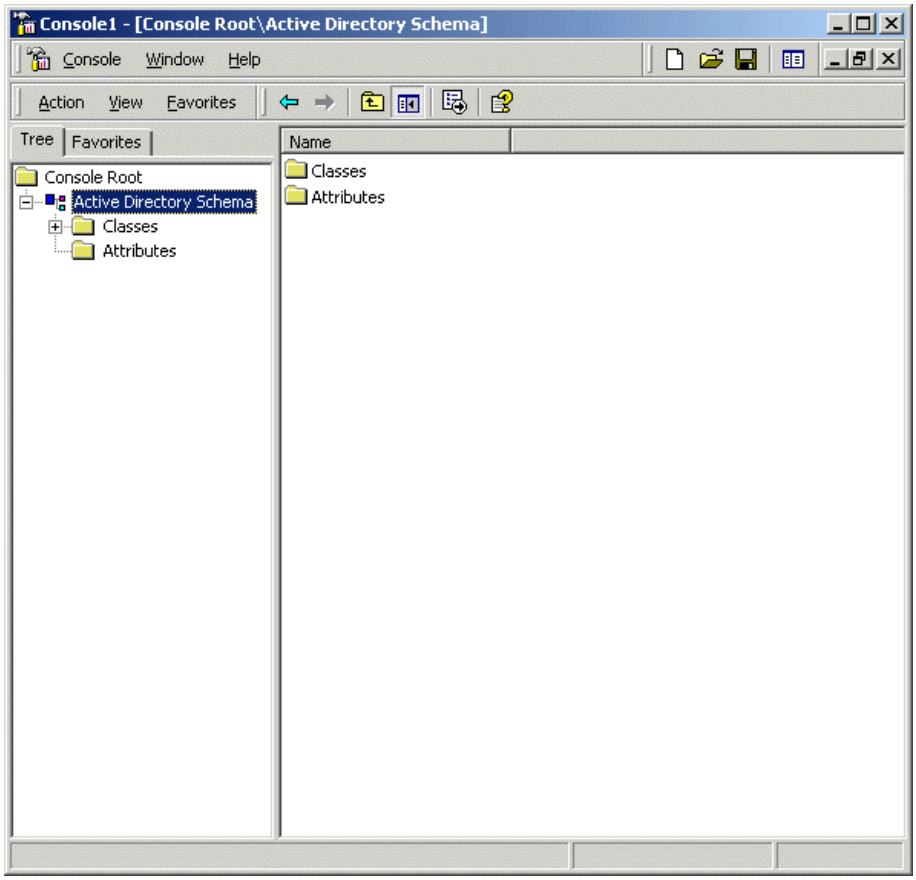

Figure 126. An Active Directory Schema MMC Console

This allows you to browse all classes and attributes. To actually modify the schema, however, you have to perform an additional step: to enable schema modifications on the Schema Operations Master.

To identify the Schema Operations Master and enable modifications to the schema, right-click **Active Directory Schema** and select **Operations Master...**, as shown in Figure 127:

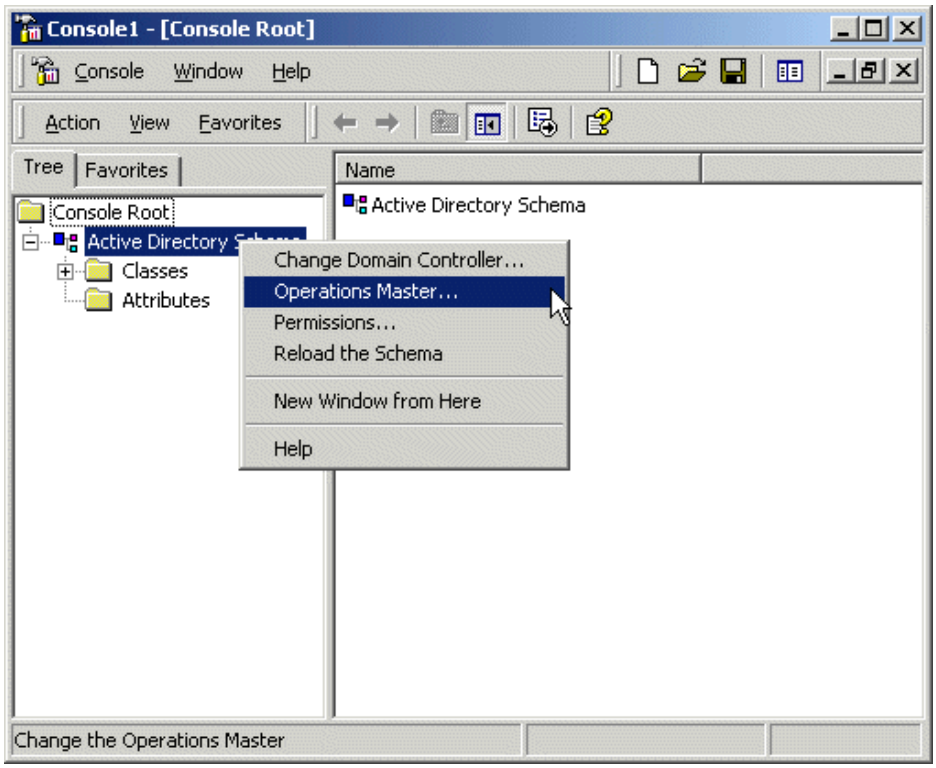

Figure 127. Identifying the Schema Operations Master

You will be presented with the Change Schema Master dialog box, as shown in Figure 128. It displays three very important pieces of information: the server you are connected to, the server that holds the Schema Operations Master role, and whether changes to the Schema are enabled on the Schema Operations Master. To allow changes to the Schema to be made, just make sure that the **The Schema may be modified on this Domain Controller** field is marked, and click **OK**.

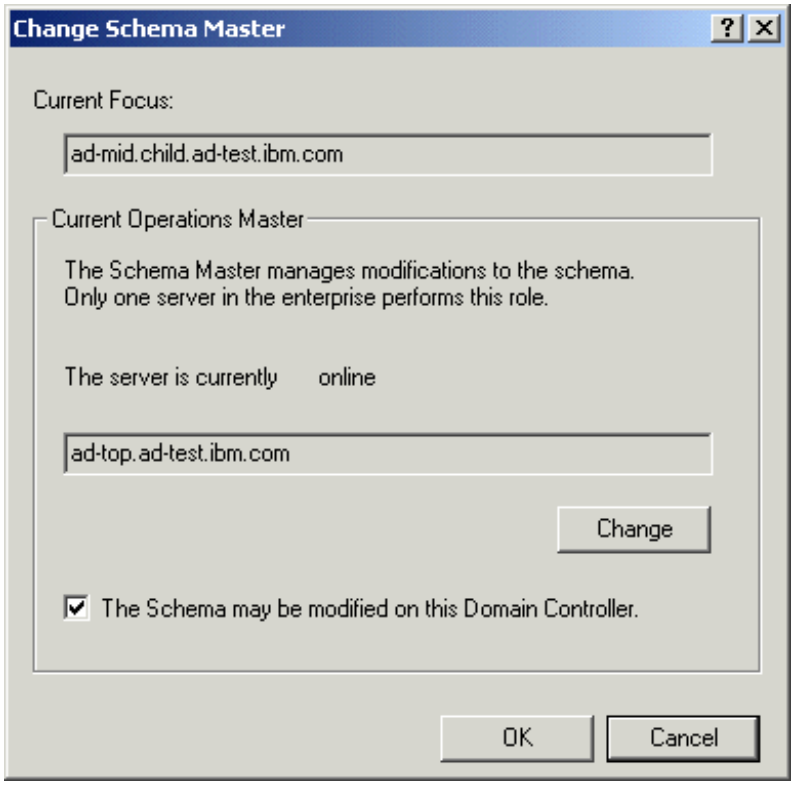

Figure 128. The Change Schema Master dialog box

Appendix D. Standards

This appendix lists the pertinent formal and informal standards (RFCs and Internet drafts) that relate to LDAP. A brief description or the actual abstract of the RFCs or the Internet Drafts, found on many Internet sites (for example, http://www.ietf.org), is added for your reference and convenience.

D.1 Core RFCs

The following is a list of the core RFCs that apply to LDAP Version 2 and Version 3 (the full text for these can be found on the IETF Web site: http://www.ietf.org):

LDAP Version 2:

- RFC 1777: Defines the LDAP Protocol
- RFC 1778: The String Representation of Standard Attribute Syntax
- RFC 1779: A String Representation of Distinguished Names
- RFC 1959: An LDAP URL Format
- RFC 1960: A String Representation of LDAP Search Filters

LDAP Version 3:

- RFC 2251: LDAP 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 LDAP V3

Other Relevant RFCs:

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

D.2 Descriptions and abstracts

This section has the description and abstracts for some of the core RFCs and some of the other RFCs and drafts that we found interesting.

- RFC 2222, Simple Authentication and Security Layer (SASL)

This describes a method for adding authentication support to connection-based protocols. To use this specification, a protocol includes a command for identifying and authenticating a user to a server and for optionally negotiating protection of subsequent protocol interactions. If its use is negotiated, a security layer is inserted between the protocol and the connection. This document describes how a protocol specifies such a command, defines several mechanisms for use by the command, and defines the protocol used for carrying a negotiated security layer over the connection.

- RFC 2251, Lightweight Directory Access Protocol (V3)

  This describes the LDAP Version 3 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. 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.

- RFC 2252, Lightweight Directory Access Protocol (V3): Attribute syntax definitions

  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 a telephone number, is encoded. It also defines matching rules to determine if values meet search criteria.

- RFC 2253, Lightweight Directory Access Protocol (V3): UTF-8 string representation of distinguished names

  This RFC defines how distinguished names are represented as strings. A string representation is easy to encode and decode and is also human readable.

- RFC 2254, The String Representation of LDAP Search Filters

  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 search filter (| (sn=Smith) (cn=Diana*)) searches for entries that either have a

surname attribute of Smith or that have a common name attribute that begins with Diana.

- 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.

- RFC 2256, A Summary of the X.500(96) User Schema for Use with LDAP V3

  Many schemas 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.

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

  The Lightweight Directory Access Protocol (LDAP) uses X.500-compatible distinguished names for providing unique identification of entries.

  This RFC defines an algorithm by which a name registered with the Internet Domain Name System (DNS) can be represented as an LDAP distinguished name.

- RFC 2589, Lightweight Directory Access Protocol (V3): Extensions for Dynamic Directory Services

  This document defines the requirements for dynamic directory services and specifies the format of request and response extended operations for supporting client/server interoperation in a dynamic directories environment. The Lightweight Directory Access Protocol (LDAP) supports lightweight access to static directory services allowing relatively fast search and update access. Static directory services store information about people that persists in its accuracy and value.

- RFC 2820, Access Control Requirements for LDAP

  This document describes the fundamental requirements of an access control list (ACL) model for the Lightweight Directory Application Protocol (LDAP) directory service. It is intended to be a gathering place for access control requirements needed to provide authorized access to and interoperability between directories.

- RFC 2829, Authentication Methods for LDAP

This document specifies particular combinations of security mechanisms that are required and recommended in LDAP implementations.

- RFC 2849, The LDAP Data Interchange Format (LDIF) - Technical Specification

  This document describes a file format suitable for describing directory information or modifications made to directory information. The file format, known as LDAP Data Interchange Format (LDIF), is typically used to import and export directory information between LDAP-based directory servers or to describe a set of changes that are to be applied to a directory.

- A simple LDAP Change Notification Mechanism <draft-ietf-ldapext-psearch-01.txt>

  This document defines two controls that extend the LDAP V3 search operation to provide a simple mechanism by which an LDAP client can receive notification of changes that occur in an LDAP server. The mechanism is designed to be very flexible yet easy for clients and servers to implement.

- LDAP V3 Security Parameters <draft-hassler-ldapv3-secparam-00.txt>

  This document defines an LDAP control called LDAPSecurityParameters for transferring security parameters with LDAP operations. With this control, it is possible to append digital signature to LDAP operations and, in this way, provide for message authenticity, message integrity, non-repudiation of message origin, and message freshness.

- LDAP V3 Triggered Search Control <draft-ietf-ldapext-trigger-01.txt>

  This document defines an LDAP V3 control to be used on the search request to allow a client to retrieve information on changes that are made to the directory information tree held by that server.

- The Java LDAP Application Program Interface <draft-ietf-ldapext-ldap-java-api-03.txt>

  This document defines a Java language application program interface to the Lightweight Directory Access Protocol (LDAP) in the form of a class library. It complements but does not replace RFC 1823, which describes a C language application program interface. It updates and replaces Draft draft-ietf-ldapext-ldap-java-api-01.txt in adding minor API extensions.

Appendix E. X.500 object identifiers (OIDs)

1.7.4.1, “Data model” on page 40 mentions every class must be uniquely identified by an OID. But what is an OID?

Every object class that is part of a schema receives a dotted decimal hierarchical identifier. These OIDs can be organized in a tree structure, very similar to the X.500 DIT (see 1.2.1.6, “The X.500 directory structure” on page 14), called Object Identifier Tree (OIT). They were defined by the X.208 standard, and they have been revised and updated by the X.680 standard.

Immediately below the root of this tree, there are only three values currently defined. You can see their meanings in Table 8.

Table 8. First-level OID values

Value	Allocated use
0	ITU-T (formerly CCITT) only
1	ISO only
2	ITU-T and ISO jointly

Below ITU (0), there are four possible values. Those are the ones defined in the X.680 annex C (ISO 8824-1:1995) standard:

Table 9. ITU first-level OID values

Value	Allocated use
0.0	ITU-T Recommendations A to Z (1 to 26)
0.1	ITU-T Questions (ITU Study group, study period and question number)
0.2	X.121 DCCs (Data Country Codes)
0.3	X.121 DNICs (Data Networks Identification Codes)
0.4	ITU-T Identified Organizations (added by X.680)
0.9	ITU-T Data

Note

The X.680 does not define OID 0.9. It is unoffically used on some RFCs, however.

ISO (1) also has four first-level values:

Table 10. ISO first-level OID values

Value	Allocated use
1.0	ISO standards (followed by the number of the standard)
1.1	ISO registration authorities (never used; retired by X.680)
1.2	ISO member-bodies (followed by the country code, as defined on the ISO 3166 standard)
1.3	ISO identified organizations (followed by the International Code Designator, as defined in the ISO 6523 standard)

Some “interesting” OIDs in this subtree include:

- 1.2.840: one of the OIDs assigned to the United States
- 1.2.840.113556 - Microsoft
- 1.3.6 - US Department of Defense
- 1.3.6.1: the Internet OID
- 1.3.18 - IBM
- 1.3.22 - Open Software Foundation
- 1.3.24 - Digital Equipment Corporation
- 1.3.26 - NATO Identified Organization
- 1.3.52 - Society of Motion Picture and Television Engineers
- 1.3.90 - Internet Assigned Numbers Authority (IANA)

Finally, the number 2 subtree is used for standards jointly defined by the ISO and the ITU. The most important, for this document’s purposes, is the Directory Standard (DS), which received the OID 2.5. Thus, all OIDs allocated by this standard start with this prefix. Some examples include:

- All user attribute types: 2.5.4
- All object classes: 2.5.6
- All matching rules: 2.5.13

The Web site `http://www.alvestrand.no/objectid/` is also an invaluable source of information on existing OIDs.

Appendix F. OS/390 samples

Following is a sample started procedure, sample environment varaibles and sample configuration file for our OS/390 environment.

F.1 Sample started procedure

```
//*************************************************************//
//* Licensed Materials - Property of IBM
//* 5647-A01
//* (C) Copyright IBM Corp. 1997, 1999
//*************************************************************//
//*************************************************************//
//*
//* Procedure for starting the LDAPSRV server
//*
//* To start server using configuration file
//* /etc/ldap/slapd.conf specify:
//*  s ldapsrv
//*
//* To start server using alternate configuration file or
//* other parameters specify:
//*  s ldapsrv,parms='options'
//* where options can be:
//*           -f filename  # alternate configuration file
//*           -d level     # debug level (65535 turns on all debugs)
//*           -p portno    # non-secure port number
//*           -s portno    # secure port number
//*
```

```
//* An alternative to the -f option is to define a CONFIG DD.
//* The remaining options are optional.  If not set, message/debug
//* levels are set to 0, non-secure port number will be 389, and
//* secure port number will be 636.  NOTE:  use of these low port
//* numbers will require that the LDAPSRV server run under a userid
//* that has OpenEdition UID 0.
//*
//* In the JCL below, GLDHLQ refers to the high level qualifier
//* that was used to install the LDAP Server datasets.  This will
//* have to be customized for the installation.
//*
//*************************************************************//
//*************************************************************//
//LDAPSRV  PROC REGSIZE=64M,
//*-------------------------------------------------------------------
//* CUSTOMIZABLE SYMBOLIC PARAMETERS
//*-------------------------------------------------------------------
//* PARMS='-f /usr/lpp/ldap/etc/slapd.conf',
//* PARMS='-f /etc/ldap/slapd3.conf',
//* PARMS='-f /etc/ldap/slapd3.conf -d 16',
//* PARMS='-f /etc/ldap/slapd3.conf -d 65535',
//* PARMS='-f /etc/ldap/slapd_r2617.conf -d 15',
//  PARMS='-f /etc/ldap/slapd_r2617.conf -d 0',
//* PARMS='-f /etc/ldap/slapd4.conf -d 65535',
//* PARMS='-f /etc/ldap/slapd2.conf',
//* PARMS='-f /etc/ldap/slapd.conf',
//  OUTCLASS='A'
```

```
//*-------------------------------------------------------------------
//GO       EXEC PGM=GLDSLAPD,REGION=&REGSIZE,TIME=1440,
//         PARM=('/&PARMS >DD:SLAPDOUT 2>&1')
//*-------------------------------------------------------------------
//* STEPLIB must be customized based on install HLQ.
//*-------------------------------------------------------------------
//STEPLIB   DD DSN=GLD.SGLDLNK,DISP=SHR
//          DD DSN=DB2V510.SDSNLOAD,DISP=SHR
//          DD DSN=DB2V510.SDSNDBRM,DISP=SHR
//          DD DSN=CEE.SCEERUN,DISP=SHR
//*-------------------------------------------------------------------
//* CONFIG can be used to specify the LDAP server config file.
//* If this DD is used, the name of the dataset must be customized
//* for the installation.
//*-------------------------------------------------------------------
//*CONFIG   DD DSN=IRIANI.SLAPD.CONF,DISP=SHR
//*-------------------------------------------------------------------
//* ENVVAR can be used to specify any environment variables
//* If this DD is used, the name of the dataset must be customized
//* for the installation.
//*-------------------------------------------------------------------
//*ENVVAR   DD DSN=IRIANI.LDAP.ENVVAR,DISP=SHR
//ENVVAR   DD DSN=TCPIP.TCPPARMS.R2617(LDAPENV),DISP=SHR
//*-------------------------------------------------------------------
//* DSNAOINI can be used to specify the dsnaoini file required by DB2.
//* Alternatively, this data set can also be specified in the
//* configuration file.
```

```
//* If this DD is used, the name of the dataset must be customized
//* for the installation.
//*--------------------------------------------------------------------
//DSNAOINI DD DSN=DB2V510.DSNJ.DSNAOINI,DISP=SHR
//SLAPDOUT DD SYSOUT=&OUTCLASS
//SYSOUT  DD SYSOUT=&OUTCLASS
//SYSUDUMP DD SYSOUT=&OUTCLASS
//CEEDUMP  DD SYSOUT=&OUTCLASS
//SYSTCPD  DD DISP=SHR,DSN=TCPIP.TCPPARMS(TCPD03A)
```

F.2 Sample environment variables

ENVVAR DD referenced in sample procedure
[TCPIP.TCPPARMS.R2617(LDAPENV)]

```
PATH=/bin:/usr/bin:/usr/lpp/ldap/sbin:.
LIBPATH=/lib:/usr/lib:.
NLSPATH=/usr/lib/nls/msg/%L/%N
MANPATH=/usr/man/C
LANG=C
LDAP_BASEDN=o=IBM_US,c=US
_BPXK_SETIBMOPT_TRANSPORT=TCPIPA
TZ=EST5EDT
```

F.3 Sample configuration file

/etc/ldap/slapd_r2617.conf referenced in sample procedure

```
#======================================================================
#  TOP SECTION
======================================================================
include         /etc/ldap/slapd.at.system
```

```
include         /etc/ldap/slapd.cb.at.conf
include         /etc/ldap/slapd.at.conf
include         /etc/ldap/slapd.oc.system
include         /etc/ldap/slapd.cb.oc.conf
include         /etc/ldap/slapd.oc.conf
include         /etc/ldap/pagent_at.conf
include         /etc/ldap/pagent_oc.conf
# ----------------------------------------
# objectclass <name> requires <nameÝ,name,..."> allows <nameÝ,name,..."> 
#
# Default Value: none
#
# Example:
#   objectclass newPerson requires cn,sn allows userpassword,telephonNumber
#   objectclass jobPerson requires cn,sn,jobTitle allows userpassword
# ----------------------------------------
# attribute <name Ý name ... " > <cis|ces|tel|bin|dn>
#  <columnName> <size> <normal|sensitive|critical>
#
# Default Value: none
#
# Example:
#   attribute jobTitle        cis     jobTitle     128    normal
#   attribute jobDescription  ces     jobDesc      1000   normal
# ----------------------------------------
#
# verifySchema <on|off>
```

```
#
# Default Value: on
#
# Example:
#   verifySchema on
#
# ----------------------------------------
verifySchema on
# ----------------------------------------
sizeLimit 500
timeLimit 3600
maxConnections 200
maxThreads 200
waitingThreads 10
# altserver ldap://otherhost.yourcompany.com:444
# referral ldap://ldap.yourcompany.com
referral ldap://9.24.105.229:389/dc=ad-test,dc=ibm,dc=com
# ----------------------------------------
adminDN "cn=LDAP Admin,o=IBM_US,c=US"
#adminPW 1234567
# ----------------------------------------
port 389
secureport 636
# sysplexgroupname LDAPGRP
# sysplexservername LDAPSVR1
# security <sslonly|ssl|none>
# security none
```

```
security ssl
# sslAuth <serverClientAuth|serverAuth>
sslAuth serverClientAuth
# sslCertificate LDAP_SERVER
# sslCipherSpecs <number>
# sslCipherspecs 16128
sslKeyRingFile /etc/ldap/ldapring.kdb
sslKeyRingFilePW r2615
sslKeyRingPWStashFile /etc/ldap/ldapring.sth
# ---------------------------------------
validateincomingV2strings yes
sendV3stringsoverV2as UTF-8
# ============================================================
#
#  Database-specific SECTIONS
#
# ============================================================
# ------------------------------------------------------------
#  SDBM-specific CONFIGURATION SETTINGS
# ------------------------------------------------------------
# database sdbm GLDBSDBM
# suffix "sysplex=sysplex1"
# extendedgroupsearching off
#   sizeLimit    350
#   timeLimit    60
# ------------------------------------------------------------
#
```

```
#  TDBM-specific CONFIGURATION SETTINGS
#
# ------------------------------------------------------
# database tdbm GLDBTDBM
# suffix "o=Your Company"
# dsnaoini /etc/ldap/db2cli.ini
#servername LOC1
# dbuserid LDAPSRV
# databasename LDAPR10
#   attroverflowsize 10000
# pwEncryption <none|crypt|md5|sha|des:keylabel>
# extendedgroupsearching <on|off>
#   sizeLimit    350
#   timeLimit    60
# readonly <on|off>
#   masterserver ldap://ldap.yourcompany.com
#   masterserverDN "cn=Master Server, o=Your Company"
#   masterserverPW secret
#   multiserver y
# ------------------------------------------------------
#  RDBM-specific CONFIGURATION SETTINGS
# ------------------------------------------------------
database rdbm GLDBRDBM
suffix "cn=localhost"
suffix "o=IBM_US,c=US"
# dsnaoini /etc/ldap/db2cli.ini
dsnaoini DB2V510.DSNJ.DSNAOINI
```

```
servername LOC1

dbuserid KAKKY

databasename LDAPDB1

tbspaceentry ENTRYTS

tbspace32k LENTRYTS

tbspace4k BP0

tbspacemutex LATTRTS

#

#adminDN "cn=LDAP Admin,o=IBM_US,c=US"

adminPW pwdadm

#

#   extendedgroupsearching off

#   sizeLimit    350

#   timeLimit    60

# readonly <on|off>

#   masterserver ldap://ldap.yourcompany.com

#   masterserverDN "cn=Master Server, o=Your Company"

#   masterserverPW secret

# multiserver y

# pwEncryption <none|crypt|md5|sha|des:keylabel>

# index <default|attname> <eqÝ,approxÝ,sub"˜"|none>

#

# index        cn           eq

# index        sn           eq

# ================================================

#  END OF CONFIGURATION FILE
```

Appendix G. How Kerberos works

(Reprinted from the redbook *The RS/6000 SP Inside Out*, SG24-5374.)

Kerberos

"Also spelled Cerberus, the watchdog of Hades, whose duty was to guard the entrance (against whom or what does not clearly appear)... It is known to have had three heads."

- Ambrose Bierce, *The Enlarged Devil's Dictionary*

Kerberos is a trusted third-party authentication system for use on physically insecure networks. It allows entities communicating over the network to prove their identity to each other while preventing eavesdropping or replay attacks. Figure 129 shows the three parties involved in the authentication. The Kerberos system was designed and developed in the 1980s by the Massachusetts Institute of Technology (MIT) as part of the Athena project. The current version of Kerberos is Version 5, which is standardized in RFC 1510, The Kerberos Network Authentication Service (V5).

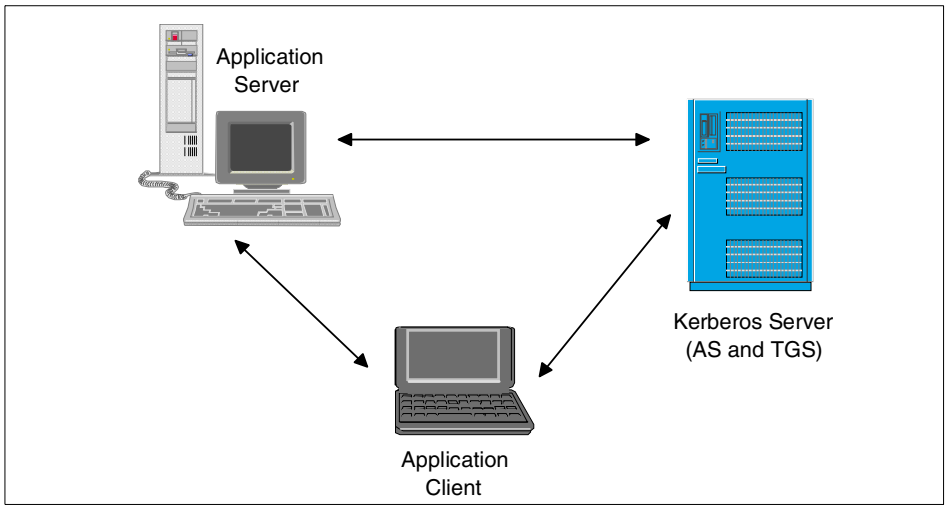

Figure 129. Partners in a third-party authentication

Kerberos provides two services to Kerberos *principals* (users or services): an Authentication Service (AS) and a Ticket-Granting Service (TGS). Principals can prove their identity to the AS by a single sign-on, and will get a

Ticket-Granting Ticket (TGT) back from the AS. When one authenticated principal (the client) wants to use services of a second authenticated principal (the server), it can get a *Service Ticket* for this service by presenting its TGT to the Kerberos TGS. The Service Ticket is then sent from the client to the server, which can use it to verify the client's identity.

This section describes the protocol that Kerberos uses to provide these services, independently of a specific implementation. A more detailed rationale for the Kerberos design can be found in the MIT article *Designing an Authentication System: a Dialogue in Four Scenes* available from the following URL: `ftp://athena-dist.mit.edu/pub/kerberos/doc/dialogue.PS`

G.1 Kerberos keys and initial setup

To encrypt the messages that are sent over the network, Kerberos uses a *symmetric* encryption method, normally the Data Encryption Standard (DES). This means that the same key is used to encrypt and decrypt a message, and consequently the two partners of a communication must share this key if they want to use encryption. The key is called *secret key* for obvious reasons: it must be kept secret by the two parties; otherwise the encryption will not be effective.

This approach must be distinguished from *public key* cryptography, which is an *asymmetric* encryption method. There, two keys are used: a *public key* and a *private key*. A message encrypted with one of the keys can only be decrypted by the other key, not by the one that encrypted it. The public keys do not need to be kept secret (hence the name "public"), and a private key is only known to its owner (it is not even to the communication partner as in the case of symmetric cryptography). This has the advantage that no key must be transferred between the partners prior to the first use of encrypted messages.

With symmetric encryption, principals need to provide a password to the Kerberos server before they can use the Kerberos services. The Kerberos server then encrypts it and stores the resulting key in its database. This key is the shared information that the Kerberos server and the principal can use to encrypt and decrypt messages they send each other. Initially, two principals who want to communicate with each other do not share a key, and so cannot encrypt their messages. But since the Kerberos server knows the keys of all the principals, it is called a *trusted third party* and can securely provide a common session key to the two parties.

Obviously, the initial passwords have to be entered securely, if possible at the console of the Kerberos server machine. They might also be generated by

the Kerberos server (especially if the principal is a host or service). In that case they must be securely transferred to the principal that stores (or remembers) them.

G.2 Authenticating to the Kerberos server

If a principal (typically a user) wants to use Kerberos services, for example because it wants to use an application service that requires Kerberos authentication, it first has to prove its identity to the Kerberos server. This is done in the following way:

A command to sign on to the Kerberos system is issued on the application client, typically `kinit` or `k4init`. This command sends an authentication request to the Kerberos server, as shown in Figure 130. This contains the type of service that is requested (here, the client wants to get service from the Ticket-Granting Service), the client's (principal's) name, and the IP address of the client machine. This request is sent in plain text. Note that the principal's password is not sent in this packet, so there is no security exposure in sending the request in plain text.

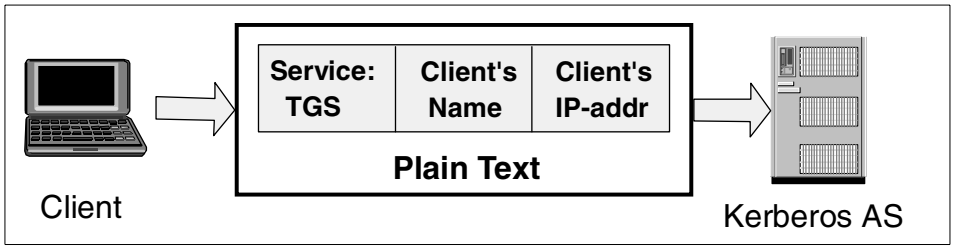

Figure 130. Client's authentication request

The request is processed by the Authentication Server. Using the client's name, it looks up the corresponding key in the Kerberos database. It also generates a random *session key* to be shared by the client and the TGS, which will be used to encrypt all future communication of the client with the TGS. With this information, the AS constructs the Ticket-Granting Ticket for the client, which (as all Kerberos tickets) contains six parts:

1. The service for which the ticket is good (here, the TGS)
2. The client's (principal's) name
3. The client machine's IP address
4. A timestamp showing when the ticket was issued

5. The ticket lifetime (maximum 21.25 hours in MIT K4, 30 days in PSSP K4, configurable in K5)
6. The session key for Client/TGS communications

This ticket is encrypted with the secret key of the TGS, so only the TGS can decrypt it. Since the client needs to know the session key, the AS sends back a reply which contains both the TGT and the session key, all of which is encrypted by the client's secret key. This is shown in Figure 131.

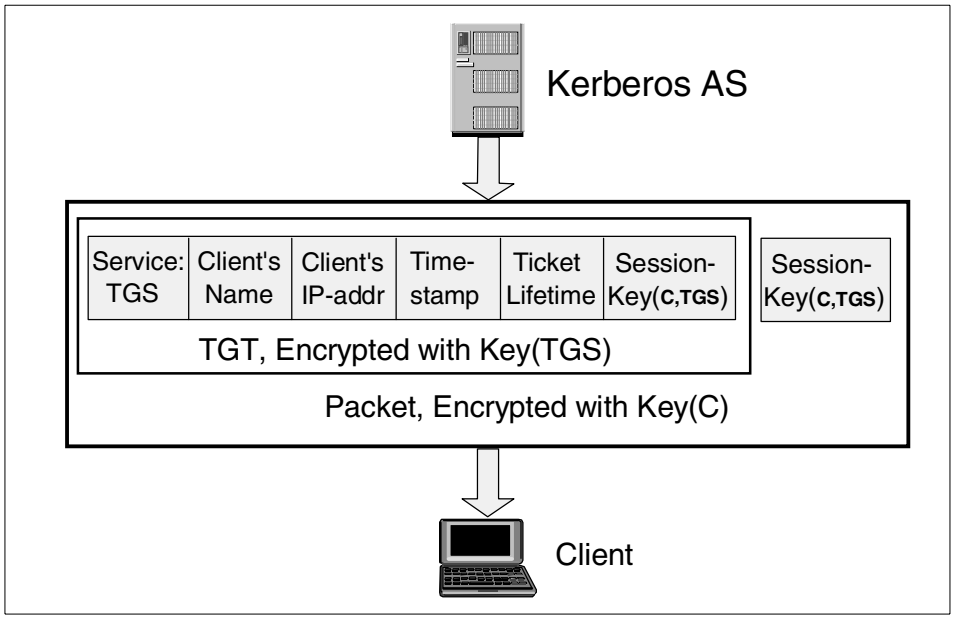

Figure 131. Authentication server's reply: TGT

Now the sign-on command prompts the user for the password, and generates a DES key from the password using the same algorithm as the Kerberos server. It then attempts to decrypt the reply message with that key. If this succeeds, the password matched the one used to create the user's key in the Kerberos database, and the user has authenticated herself. If the decryption failed, the sign-on is rejected and the reply message is useless. Assuming success, the client now has the encrypted Ticket-Granting Ticket and the session key for use with the TGS, and stores them both in a safe place. Note that the authentication has been done locally on the client machine, the password has not been transferred over the network.

G.3 Authenticating to an application server

If the client now wants to access an application service that requires Kerberos authentication, it must get a service ticket from the Ticket-Granting Service. The TGT obtained during the Kerberos sign-on can be used to authenticate the client to the TGS; there is no need to type in a password each time a service ticket is requested.

If the client sent only the (encrypted) TGT to the Kerberos TGS, this might be captured and replayed by an intruder who has impersonated the client machine. To protect the protocol against such attacks, the client also generates an *authenticator* which consists of three parts:

1. The client's (principal's) name
2. The client machine's IP address
3. A timestamp showing when the authenticator was created

The authenticator is encrypted with the session key that the client shares with the TGS. The client then sends a request to the TGS consisting of the name of the service for which a ticket is requested, the encrypted TGT, and the encrypted authenticator. This is shown in Figure 132.

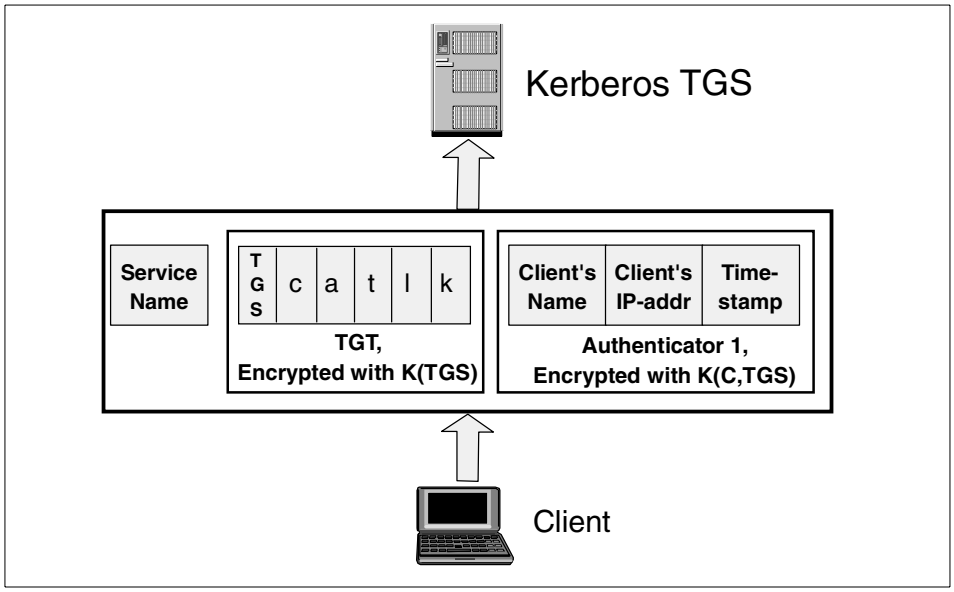

Figure 132. Client's service ticket request

The Ticket-Granting Server can decrypt the TGT since it is encrypted with its own secret key. In that ticket, it finds the session key to share with the client. It uses this session key to decrypt the authenticator, and can then compare the client's name and address in the TGT and the authenticator.

If the timestamp that the TGS finds in the authenticator differs from the current time by more than a prescribed difference (typically 5 minutes), a ticket replay attack is assumed and the request is discarded.

If all checks pass, the TGS generates a service ticket for the service indicated in the client's request. The structure of this service ticket is identical to the TGT described in G.2, "Authenticating to the Kerberos server" on page 193. The content differs in the service field (which now indicates the application service rather than the TGS), the timestamp, and the session key. The TGS generates a new, random key that the client and application service will share to encrypt their communications. One copy is put into the service ticket (for the server), and another copy is added to the reply package for the client since the client cannot decrypt the service ticket. The service ticket is encrypted with the secret key of the service, and the whole package is encrypted with the session key that the TGS and the client share. The resulting reply is shown in Figure 133. Compare this to Figure 131 on page 194.

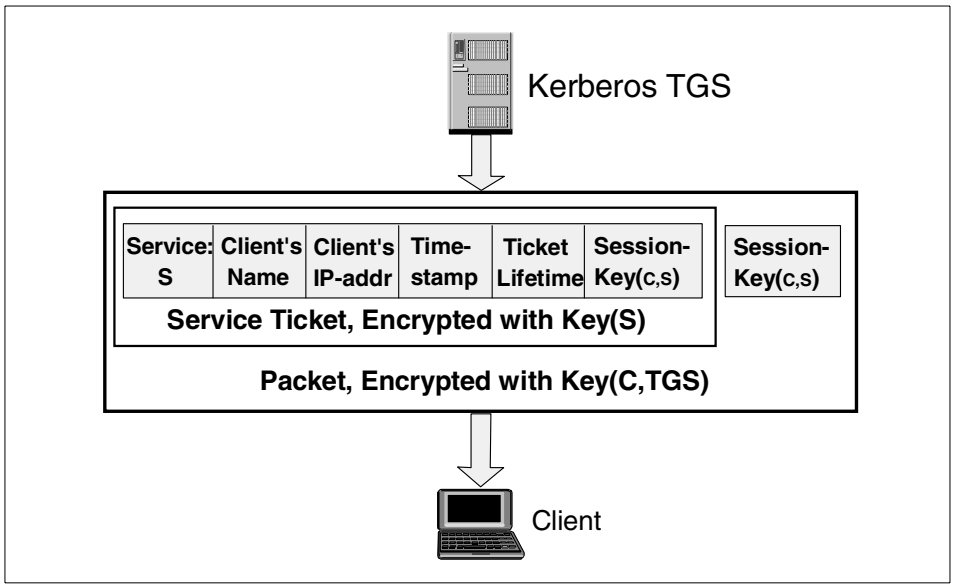

Figure 133. Ticket granting service's reply: service ticket

The client can decrypt this message using the session key it shares with the TGS. It then stores the encrypted service ticket and the session key to share with the application server, normally in the same *ticket cache* where it already has stored the TGT and session key for the TGS.

To actually request the application service, the client sends a request to that server which consists of the name of the requested service, the encrypted service ticket, and a newly generated authenticator to protect this message against replay attacks. The authenticator is encrypted with the session key that the client and the service share. The resulting application service request is shown in Figure 134. Note the resemblance to the request for Ticket-Granting Service in Figure 132 on page 195.

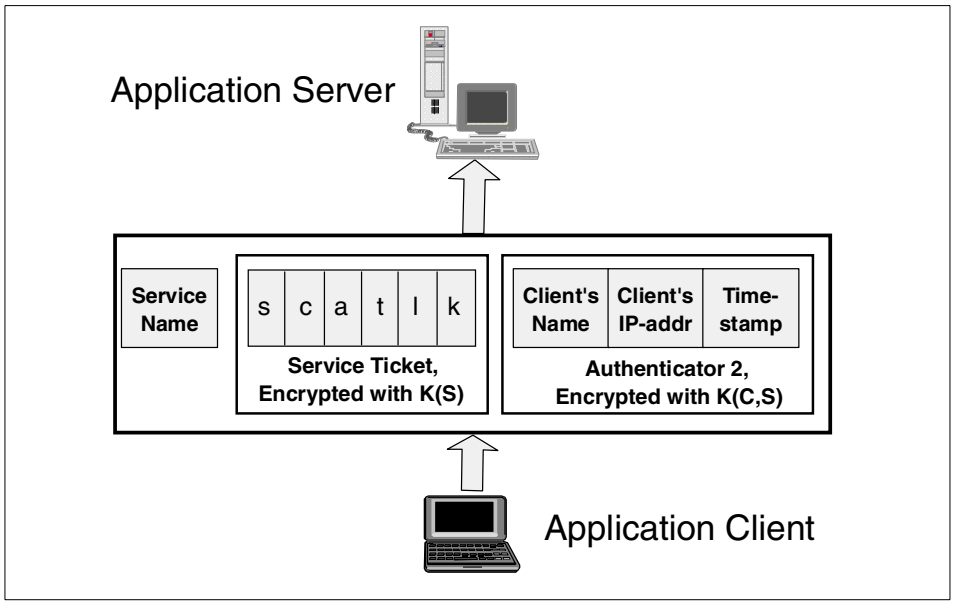

Figure 134. Client's application service request

The application server decrypts the service ticket with its secret key, uses the enclosed session key to decrypt the authenticator, and checks the user's identity and the authenticator's timestamp. Again, this processing is the same as for the TGS processing the service ticket request. If all checks pass, the server performs the requested service on behalf of the user.

Authorization

Kerberos is only responsible for authenticating the two partners. Any authorization mechanism must be enforced by the application itself.

If the client required mutual authentication (that is, the service has to prove its identity to the client), the server could send back a message which is encrypted by the session key it shares with the client, and an application-dependent contents that the client can verify. Since the service can only know the session key if it was able to decrypt the service ticket, it must have known its secret key and so has proven its identity.

Appendix H. Modifying the Domino schema

This appendix shows how to export and modify the Lotus Domino LDAP schema. These procedures would be used for extending the Lotus Domino schema in a similar fashion as was done in Chapter 5, “Scenario4: Extending schemas and synchronizing data” on page 121.

H.1 Exporting the LDAP schema

To export the LDAP schema to a Notes database, type `tell LDAP exportschema` from the Domino server console. This will create a database, Schema50.nsf, in the root of the data directory. This database has an entry for every item in the Domino schema and how it maps to the LDAP standard. See Figure 135 for an example of an entry in the schema database.

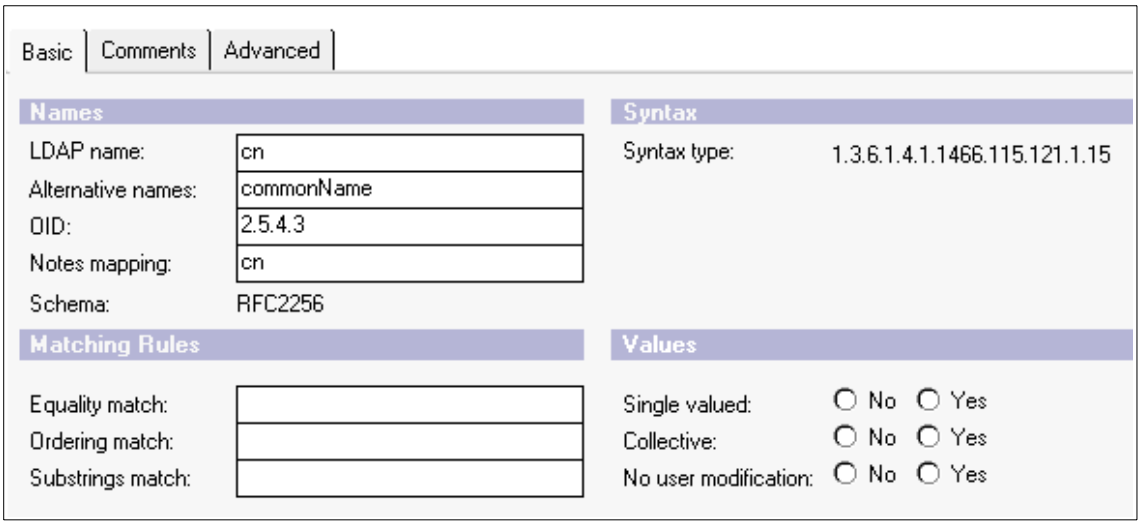

Figure 135. Document in Schema50.nsf

Here you can see the name of the attribute (cn in this case), any alternative name defined in the schema, the object’s OID, the Notes field name that is mapped to that attribute, and the schema from which the attribute is derived.

Note that this database contains Notes field names for attributes that aren’t included on any form, such as carLicense. You should check this database before extending the schema, and use the Notes mappings whenever possible before creating your own attributes.

H.2 Modifying the LDAP schema

Since the LDAP schema for Domino directory services is simply an extension of the inherent design of the Domino directory, to modify the schema you simply modify that design. All of the design changes are made using the Domino designer client, and are thus made available to both Notes and LDAP clients.The procedure for extending the schema varies slightly depending upon whether you are adding new attributes to the existing schema (person, group, etc.) or adding totally new object classes to the schema.

H.2.1 Adding new attributes to the existing schema

To ease the addition of new design elements to the existing schema for People, Groups, etc., Lotus has added to the Domino directory a set of subforms that may be modified to include user-defined fields. These subforms are blank placeholders that can be used to add attributes to the relevant object classes without the risk of losing those changes when the design of the directory is refreshed. The subforms included in the Domino directory template are:

- $CertifierExtensibleSchema
- $DomainExtensibleSchema
- $GroupExtensibleSchema
- $MailInDatabaseExtensibleSchema
- $PersonExtensibleSchema
- $ResourceExtensibleSchema

You will note that there is not a $ExtensibleSchema subform for servers. If you find it necessary to extend the server schema, you should add your own subform to the server form. Make sure that the server form is set to “Disallow design refresh/replace to modify” so that your change won’t be lost. Then add your fields onto the subform. When you upgrade to a newer release and have to update the design of the server form, you should be able to easily reapply your changes in the same way.

We recommend that you create a new subform to store your own extensions, and then insert that subform into the relevant $extensibleschema subform. You can then make all of your changes in one place. As an example, let’s investigate how you would add a new attribute for eye color to the person object class.

Your first step should be to back up your copy of the Domino Directory template, in case anything goes wrong. Next, open Domino Designer and click **Open an existing database...** From the dialog that results, choose the server and the Domino directory template database (**pubnames.ntf**). If you have a multiple-server environment, make sure that you pick a Domino server that can push your design changes to the other servers. Your administrative hub server is probably the most logical choice.

Now, create a new subform. From the Design pane, expand the **Resources** section. Click **Subforms**. Now click **New Subform**. You should be presented with a blank subform. Your first step is to define this subform. Since it is a custom extension of the Person object, we will call our subform “Acme Person.” Choose **Create->Field**. A new field, temporarily named “Untitled” should be created for you on the form, and the Field properties box should appear. Change the field name to **AcmePerson**. Now set the field type to “Text, Computed when composed.” Close the dialog and move to the programmer’s pane. For the field’s value, enter this formula:

```
"FIELD $objectclass:=$objectclass:"AcmePerson";1"
```

This sets up AcmePerson as an auxiliary object class to dominoPerson.

Now you can add any custom fields that you want. In our example, we are extending the schema to include an attribute for eye color, so position the cursor wherever you want the new field to appear in Notes, and choose **Create->Field**. For our example, we’ll name the field “EyeColor”. This field will be a simple text field. The type of field you create will translate to the type of attribute in the schema displayed in Table 11.

Table 11. Field to schema mapping

Notes Field Type	Schema Attribute Syntax
Text	Directory string
Date/Time	Generalized time
Number	Integer
Names	Distinguished name

This field is a normal Notes field, so you can apply any input validation/translation formulas to it that you wish. Once you’ve added the field, close and save the subform. If prompted, give it a name of “Acme Person”.

The final step in adding this element to the person object class is to embed the custom subform that we’ve created (AcmePerson) into the

$PersonExtensibleSchema subform that Lotus has set up for exactly this purpose. To do this, double-click the **$PersonExtensibleSchema** subform and choose **Create->Insert Subform**. Now choose the newly-created subform from the dialog that pops up. This will insert it into the $PersonExtensibleSchema subform. Now close and save the $PersonExtensibleSchema subform.

From Notes, the extra field that you've added will appear in the Person document under the Other tab. If you have a lot of custom data to add to the person form, you could embed a tabbed table containing all of the different fields. You can also add any embellishments to the Notes GUI that you wish in order to make the appearance better. LDAP clients, of course, will not see the appearance, only the underlying structure.

Finally, to recognize the new changes that you've made to the Person object class, you need to reload the schema into the LDAP server. To do this, simply type `tell ldap reloadschema`. This will update the schema with the changes you have made. If you want to verify that your changes have made it into the schema, you can type `tell ldap exportschema` and examine the resulting schema50.nsf database that is created on the directory server.

You can follow those same instructions to extend the other portions of the Domino LDAP schema.

H.2.2 Adding new object classes to the schema

To add an entirely new object class to the schema, you need to create a new form in the Domino directory. Designing forms is basically the same as designing subforms, but they are stand-alone entities. You won't have to insert them into another design element to make them work.

You will, however, have to add two special fields to the form that you are creating. These will tell Domino where to add the object class into the schema. Once you've created a new, blank form, add a field named Fullname, and set its type to Names. This field will store the Distinguished name of the entry. The second field to add should be called "type", and it should be text, computed when composed. For its formula, use the name of the form. That is, if you are adding a new object class to define printers, for example, name the form "Printer" and give the field "Type" the formula "Printer". The new object class that you define will be given the value of the field "Type" and will exist in your attribute tree under "top".

Now you can add any additional fields to the form. Whenever possible, use the standard LDAP field names from the schema50.nsf database. Use the

data type mappings from Table 11 on page 201 to help translate Notes fields to LDAP attributes.

Once you've added all of your fields, save the form. You will probably want to create a view in the Domino directory to display these documents.

Appendix I. X.509 certificates

Although there have been several proposed formats for public key certificates, most commercial certificates available today are based on the international standard ITU-T Recommendation X.509 (formerly CCITT X.509).

X.509 certificates are used in secure Internet protocols such as:

- Secure Socket Layer (SSL)
- Secure Multi-purpose Internet Mail Extension (S/MIME)
- Secure Electronic Transaction (SET)

I.1 The X.509 standard

Originally, the X.509 standard was intended to specify the authentication service for X.500 directories. Directory authentication in X.509 can be done using either secret-key techniques or public-key techniques. The latter is based on public-key certificates. At present, the public-key certificate format defined in the X.509 standard is widely used and supported by a number of protocols in the Internet world. X.509 standard does not specify a particular cryptographic algorithm; however, apparently RSA algorithm is the most broadly used one.

The initial version of X.509 was published in 1988. The public-key certificate format defined in this standard is called X.509 Version 1 (X.509v1). When X.500 was revised in 1993, two more fields were added, resulting in the X.509 Version 2 (X.509v2) format.

X.509 Version 3 (X.509v3) was proposed in 1994. X.509v3 extends v2 in order to address some of the security concerns and limited flexibility that were issues in Versions 1 and 2. The major difference between Versions 2 and 3 is the addition of the extensions field. This field grants more flexibility since it can convey additional information beyond just the key and name binding. In June 1996, standardization of the basic v3 format was completed.

I.2 X.509 certificate content

An X.509 certificate consists of the following fields:

- Version of certificate format
- Certificate serial number

- Digital signature algorithm identifier (for issuer's digital signature)
- Issuer name (that is, the name of the Certificate Authority)
- Validity period Subject (that is, user or server) name
- Subject public-key information: algorithm identifier and public-key value
- Issuer unique identifier, Versions 2 and 3 only (added by Version 2)
- Subject unique identifier, Versions 2 and 3 only (added by Version 2)
- Extensions, Version 3 only (added by Version 3)
- Digital signature by issuer on the above fields

Standard extensions include subject and issuer attributes, certification policy information, and key usage restrictions, among others.

Appendix J. LDAP utilities

This appendix lists the main Microsoft and IBM LDAP command-line utilities. Some of these utilities (especially `ldifde`) have been extensively used throughout this book for exporting, importing and modifying directory and schema information.

J.1 ldifde utility

LDIF Directory Exchange (LDIFDE) is Microsoft's LDIF utility. Following is a listing of the switches that may be used with this utility.

```
LDIF Directory Exchange


General Parameters

==================

-i              Turn on Import Mode (The default is Export)

-f filename     Input or Output filename

-s servername   The server to bind to (Default to DC of logged in Domain)

-c FromDN ToDN  Replace occurences of FromDN to ToDN

-v              Turn on Verbose Mode

-j              Log File Location

-t              Port Number (default = 389)

-u              Use Unicode format

-?              Help


Export Specific

===============

-d RootDN       The root of the LDAP search (Default to Naming Context)

-r Filter       LDAP search filter (Default to "(objectClass=*)")

-p SearchScope  Search Scope (Base/OneLevel/Subtree)

-l list         List of attributes (comma separated) to look for
                in an LDAP search

-o list         List of attributes (comma separated) to omit from
                input.
```

```
-g              Disable Paged Search.
-m              Enable the SAM logic on export.
-n              Do not export binary values

Import
======
-k              The import will go on ignoring 'Constraint Violation'
                and 'Object Already Exists' errors
-y              The import will use lazy commit for better performance

Credentials Establishment
=========================
Note that if no credentials is specified, LDIFDE will bind as the currently
logged on user, using SSPI.

-a UserDN [Password | *]            Simple authentication
-b UserName Domain [Password | *]   SSPI bind method

Example: Simple import of current domain
    ldifde -i -f INPUT.LDF

Example: Simple export of current domain
    ldifde -f OUTPUT.LDF

Example: Export of specific domain with credentials
    ldifde -m -f OUTPUT.LDF
           -b USERNAME DOMAINNAME *
           -s SERVERNAME
           -d "cn=users,DC=DOMAINNAME,DC=Microsoft,DC=Com"
           -r "(objectClass=user)"
```

J.2 ldapmodify utility

ldapmodify is an IBM client utility that sends modify or add requests to an LDAP server.

```
usage: ldapmodify [options] [-f file]

where: file: name of input file

note: standard input is used if file is not specified

options:
-h host      LDAP server host name
-p port      LDAP server port number
-D dn        bind dn
-w password  bind password
-Z           use a secure ldap connection (SSL)
-K keyfile   file to use for keys
-P key_pw    keyfile password
-N key_name  private key name to use in keyfile
-m mechanism perform SASL bind with the given mechanism
-R           do not chase referrals
-M           Manage referral objects as normal entries.
-O maxhops   maximum number of referrals to follow in a sequence
-V version   LDAP protocol version (2 or 3; default is 3)
-C charset   character set name to use, as registered with IANA
-a           force add operation as default
-r           force replace operation as default
-b           support binary values from files (old style paths)
-c           continuous operation; do not stop processing on error
-n           show what would be done but don't actually do it
-v           verbose mode
-d level     set debug level in LDAP library
```

J.3 ldapadd utility

ldapadd is an IBM client utility that sends modify or add requests to an LDAP server. ldapadd is implemented as a renamed version of ldapmodify. When invoked as ldapadd the -a (add new entry) flag is turned on automatically.

```
usage: ldapadd [options] [-f file]

where: file: name of input file

note: standard input is used if file is not specified

options:
-h host      LDAP server host name
```

```
-p port      LDAP server port number
-D dn        bind dn
-w password  bind password
-Z           use a secure ldap connection (SSL)
-K keyfile   file to use for keys
-P key_pw    keyfile password
-N key_name  private key name to use in keyfile
-m mechanism perform SASL bind with the given mechanism
-R           do not chase referrals
-M           Manage referral objects as normal entries.
-O maxhops   maximum number of referrals to follow in a sequence
-V version   LDAP protocol version (2 or 3; default is 3)
-C charset   character set name to use, as registered with IANA
-a           force add operation as default
-r           force replace operation as default
-b           support binary values from files (old style paths)
-c           continuous operation; do not stop processing on error
-n           show what would be done but don't actually do it
-v           verbose mode
-d level     set debug level in LDAP library
```

J.4 ldapdelete utility

ldapdelete is an IBM client utility that opens a connection to an LDAP server and binds and deletes one or more entries. If one or more DN arguments are provided, entries with those Distinguished Names (DN) are deleted. Each DN should be 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 -f flag is used).

J.5 ldapsearch utility

ldapsearch is an IBM client utility that opens a connection to an LDAP server and binds and performs a search using the filter *filter*. The *filter* should conform to the string representation for LDAP filters (RFC-1558 compliant).

If ldapsearch finds one or more entries, the attributes specified are retrieved and the entries and values are printed to standard output. If no attributes are given, all attributes are returned.

J.6 ldapmodrdn utility

ldapmodrdn is an IBM client utility that opens a connection to an LDAP server and binds and modifies the RDN of entries, effectively renaming them. The

entry information is read from standard input, from a file, through the use of the - f option, or from the command-line pair DN and RDN.

Note

IBM SecureWay also has Server Utilities, such as:

ldif
ldif2db
db2ldif
bulkload
IBM Key Management Utility (GSK4IBM)

For information on these utilities see the IBM SecureWay's help menus.

Appendix K. Special notices

This redbook is intended to help systems and network administrators, support personnel and consultants who are faced with today‘s directory management challenges in using LDAP for directory integration. The information in this publication is not intended as the specification of any programming interfaces that are provided by the discussed products. See the PUBLICATIONS section for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers

attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

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

IBM ®	Redbooks
AIX	Redbooks Logo 
AS/400	RS/6000
DB2	S/390
IBM	SecureWay
IBM.COM	SP
Netfinity	SP1
OpenEdition	System/390
OS/2	Lotus
OS/390	Lotus Notes
OS/400	Domino
RACF	Notes

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

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

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

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel

Corporation in the United States and/or other countries.

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

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Appendix L. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

L.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 219.

- *Exploiting RS 6000 SP Security: Keeping It Safe*, SG24-5521
- *Understanding LDAP*, SG24-4986
- *LDAP Implementation Cookbook*, SG24-5110
- *Security in OS390-based TCPIP Networks*, SG24-5383
- *OS390 Security Server 1999 Updates Technical Presentation Guide*, SG24-5627
- *Getting the Most From Your Domino Directory*, SG24-5986
- *Understanding IBM SecureWay FirstSecure*, SG24-5498
- *OS/390 Security Server 1999 Updates: Installation Guide*, SG24-5629
- *The RS/6000 SP Inside Out*, SG24-5374

L.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at ibm.com/redbooks for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
IBM System/390 Redbooks Collection	SK2T-2177
IBM Networking Redbooks Collection	SK2T-6022
IBM Transaction Processing and Data Management Redbooks Collection	SK2T-8038
IBM Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
IBM AS/400 Redbooks Collection	SK2T-2849
IBM Netfinity Hardware and Software Redbooks Collection	SK2T-8046
IBM RS/6000 Redbooks Collection	SK2T-8043
IBM Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

L.3 Other resources

These publications are also relevant as further information sources:

- *OS/390 Security Server LDAP Server Administration and Usage Guide*, SC24-5861

L.4 Referenced Web sites

These Web sites are also relevant as further information sources:

- http://www.alvestrand.no/objectid/
- http://www.s390.ibm.com
- http://www.dmtf.org
- http://www.ibm.com/software/enetwork/directory
- http://www.oasis.org
- http://www.ietf.org
- http://www.lotus.com
- http://w3.itso.ibm.com
- http://w3.ibm.com

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** ibm.com/redbooks

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number _____

☐ Credit card number _____

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

List of abbreviations

ACL	Access Control List
API	Application Programming Interface
ASN	Abstract Syntax Notation
CA	Certificate Authority
CCITT	Comite Consultatif International Telephonique et Telegraphique
CDS	Cell Directory Service (DCE)
CRAM-MD5	Challenge-Response Authentication Mechanism - Message Digest 5
DAP	Directory Access Protocol (X.500)
DARPA	Defense Advanced Research Projects Agency
DAS	Directory Assistance Service
DCE	Distributed Computing Environment
DEN	Directory Enabled Networks
DES	Data Encryption Standard
DIT	Directory Information Tree
DN	Distinguished Name
DNS	Domain Name System
DSA	Directory Services Agent
DSS	Directory & Security Services
DTS	Distributed Time Service
EDI	Electronic Data Interchange
EJB	Enterprise Java Beans
FTP	File Transfer Protocol
GDA	Global Directory Agent
GDS	Global Directory Service
GSO	Global Sign-On
GSSAPI	Generic Security Service API
HTTP	Hypertext Transport Protocol
IAB	Internet Architecture Board
IANA	Internet Assigned Numbers Authority
IBM	International Business Machines Corporation
IETF	Internet Engineering Task Force
IESG	Internet Engineering Steering Group
ISI	Information Sciences Institute
ISO	International Standards Organization
ISOC	Internet Society
ITSO	International Technical Support Organization
ITU-T	International Telecommunications Union - Telecommunications

JDAP Java Directory Access Protocol (context: Java LDAP Application Programming Interface)

JDBC Java Database Connectivity

JNDI Java Naming and Directory Interface (Sun)

LAN Local Area Network

LDAP Lightweight Directory Access Protocol

LDIF LDAP Data Interchange Format

MIME Multipurpose Internet Mail Extensions

NDS Novell Directory Services

NOS Network Operating System

NSI Name Service Interface (DCE)

OSF Open Software Foundation

OSI Open Systems Interconnection

RDN Relative Distinguished Name

RFC Request for Comment

RPC Remote Procedure Call

SASL Simple Authentication and Security Layer

SDK Software Development Kit

SPI Service Provider Interface

SQL Structured Query Language

SSL Secure Sockets Layer

TCP/IP Transmission Control Protocol/Internet Protocol

TLS Transport Layer Security

TME Tivoli Management Environment

UMICH University of Michigan

URL Uniform Resource Locator

WAN Wide Area Network

Index

Symbols

A

B

C

D

M

N

O

P

R

S

T

U

V

W

X

Y

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 845 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-6163-00 Using LDAP for Directory Integration: A Look at IBM SecureWay Directory, Active Directory and Domino
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. **ibm.com**/privacy/yourprivacy/

(0.2”spine)
0.17”<->0.473”
90<->249 pages

Using LDAP for Directory Integration

A Look at IBM SecureWay Directory, Active Directory and Domino

LDAP integration guidelines for systems administrators

Referrals and schema extensions

Examples to help you integrate directories

This redbook is intended for systems and network administrators, support personnel and consultants who are faced with today's directory management challenges.

This book explains directory and LDAP concepts, and deals with some basic and advanced LDAP tasks, such as referrals and schema extensibility. We demonstrate ways to tackle these issues by taking a scenario-based approach, using some leading directory products available: the IBM SecureWay Directory, Lotus Domino, and Microsoft's Active Directory.

This redbook will help you if:
- You have an LDAP Directory and you are looking to implement an Enterprise Directory.
- You have Active Directory and you looking to implement, integrate or move to the IBM SecureWay Directory.
- You have Lotus Domino and you want to leverage Domino's directory as a base for populating other directories.
- You have an Enterprise Directory and you are looking to add Active Directory to your environment.
- You want to exchange, replicate or synchronize data between the IBM SecureWay Directory and Active Directory.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6163-00 ISBN 073842112X