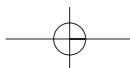
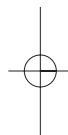
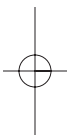
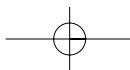
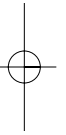
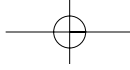
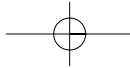


Part I

How LDAP Works







1

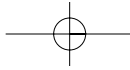
Overview of LDAP

Introducing Directories

Directories are designed to help people find their way. We've all entered an unfamiliar building and used the building's directory. Without the directory, we'd have to wander the building in search of our destination. We rely on that directory without thinking much about it, unless the information leads us to the wrong place.

With the advent of computers, there is no end of information that needs organizing so people can easily find it. Computers have always relied on directories. Even early operating systems such as DOS had a file directory so a user could keep track of data files. Directories seem to be everywhere online today, with directories that list contact information for high school graduating classes, directories that list all the movies showing, and so on. All directories have the same goal of helping us eliminate aimless searching for the information we seek.

Directories help people by organizing information



4 Overview of LDAP

Directories allow data to be managed

However, a directory should be more than just an efficient way to find information; it should also provide an efficient means of managing that information. If there are many sources for the information we seek, we may get contradictory or out-of-date information, and sifting through can be just as frustrating as aimless browsing. The directory should be a centrally managed repository. It's important to have a single, authoritative source for a particular type of information. That way, we don't have to search in several places for the information we want, and then painstakingly decide which information is correct.

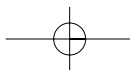
Many applications and services can take advantage of data that is centralized in a directory

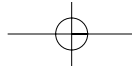
There are many uses for a directory, beyond the direct interaction a person has when manually looking up information. Application software can leverage the information in a directory to provide a more informed and better experience. Backroom services that work without our being aware of them can also make use of centralized information. These services provide the foundation that lets us interact in the digital world, identifying us to others, establishing our authority, allowing us to communicate with each other, even protecting us. Each of these foundational services, sometimes called *infrastructure*, must either have its own source of information about identities or rely on a common set of information. Clearly there is a benefit to having only a single set of information to manage, along with a clearly defined method of accessing this data. And there are many uses for the same piece of data, as the example that follows shows.

The directory can streamline your business processes

A directory should enable an organization to manage its business processes better. Imagine the following scenario as an example of why directories are making such an impact.

An important new executive joins your place of business. On her first day, the security officer stops her at the front door to request a long list of information for her security badge. Once she has passed by the security officer, her first visit is to the HR department, where she is asked to fill out a form with her





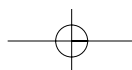
name, social security number, birth date, home address, department, supervisor, and so forth so she can be added to the payroll system. Then she is shown to her office. There a young technician gives her a user account and password for accessing network resources. The technician needs her name and department to give her access to the appropriate network resources.

Throughout the day, administrative assistants stop by for information. One needs to take down the asset information for her new computer and assign it to her by name. Another is from the HR department again with a form for benefits. Another is from the budget department, to give her the proper budget codes for requisitions and spending accounts. The forms don't seem to stop . . . and much of the information is requested on multiple forms. Naturally the executive wonders why all these people can't share her information. Ideally, she would enter the information into the directory and then other people who needed the information could query the directory without wasting her valuable time. The people performing these business tasks could manually query the directory for the appropriate information, or better yet software could be used to interact directly with the directory and automate the entire process after the executive entered the personal data.

There are as many uses for a directory as there are types of information to organize. The amount of information being stored on computers is increasing at an exponential rate, so finding a good directory solution has become more important than ever. Fortunately for the computer industry, a common standard for directories has emerged in LDAP. This chapter introduces LDAP, highlights its capabilities, and explains why it has garnered widespread support as the best directory solution.

The LDAP standard has been widely accepted as the ideal solution

To this point, I have discussed directories through common examples in everyday experience. Now it is time to look at what a



My Company Won't Buy a Directory

Maybe it should. The potential savings over the long run are more substantial than you think. For example, think of all the business processes that are keyed to correct and up-to-date contact information. When my contact information changed recently, I notified all the companies with which I did business. But I still had a difficult time because many businesses didn't use a single, unified repository for tracking that information. In some cases, I stopped doing business with them because I didn't appreciate spending my time troubleshooting their poor business process.

On another track, your company may just as easily end up with a directory because it is a required component for implementing some other essential product. Directories are becoming a common prerequisite. For example, almost all network operating systems require a directory to get the most out of product features. A lot of server software requires a directory to store its configuration information. So even if your company wouldn't buy a directory to actively solve a business need, you will probably end up with one.

directory is, and what is unique about the directory structure that makes it useful. This examination focuses on two properties:

- **Structure**—How does a directory store information?
- **Content and usefulness**—What can be put in a directory, and why would someone choose a directory over something else?

This general examination of directories sets the stage for the following introduction to LDAP.

Structure

A directory is composed of entries. The *entry* is the basic unit of the directory. These entries usually contain a similar kind of

The entry is the unit of the directory

information. For example, my directory could have entries about people (commonly called person entries) that include a person's name along with a phone number, and perhaps other relevant personal information. There would be an entry for each person, and each entry would consist of all the personal information known by the directory about that person. The term "entry" is synonymous with the term *record* or *directory object*; these terms are used interchangeably in the literature on the subject.

The information associated with an entry is called the *attributes* or *properties* of the entry. Again, the literature is not uniform; "attribute" and "property" are used interchangeably. An entry is essentially a collection of attributes. For a person entry, the person's name is one of the attributes, as is the phone number. Depending on how the directory is defined, entries can have a set of mandatory attributes as well as a set of optional attributes. For example, my directory might have entries with mandatory common name (full name) and surname (last name) attributes along with optional phone number, fax number, and e-mail address attributes. The entry is incomplete, and therefore not allowed, without the presence of every mandatory attribute. Figure 1-1 shows an example entry for myself.

The entry is composed of a set of attributes

Each attribute is composed of a pair of elements. The *attribute type* is a label for the kind of information being stored. The

The attribute is composed of a type and value pair

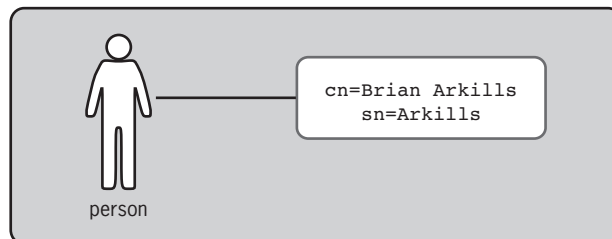


Figure 1-1 A person entry with two attributes

8 Overview of LDAP

attribute value is the actual data being stored. For example, `cn=Brian Arkills` is an attribute pair, where `cn` (or common name) is the attribute type, and `Brian Arkills` is the attribute value. Incidentally, some attributes can have multiple values, which is an important feature for maximizing the flexibility of the data structure. The ability to have multiple values is a key advantage that LDAP possesses over common database solutions. Figure 1-2 shows an entry with a multivalued `cn` attribute.

The `objectclass` attribute defines what rules the entry follows

There is a special attribute that is mandatory to all entries, called the `objectclass` attribute. This attribute determines what rules the entry follows. These rules govern the content of the entry by specifying the set of attributes that are mandatory and another set that is optional. The `objectclass` attribute is multivalued, so the set of mandatory and optional attributes for an entry is the union of all the values of the `objectclass` attribute. The rules may also include the possibility of restrictions on where entries of that object class can be created. At the most basic level, the object class defines what attributes can be used in the entry. The schema of the directory determines which object classes are available in the directory. The *schema* essentially defines the set of rules the directory data must follow.

Many types of entries are possible

A directory can have many different types of entries. A directory can have person entries with name attributes, phone attributes, and others. But it can also have entries that represent products

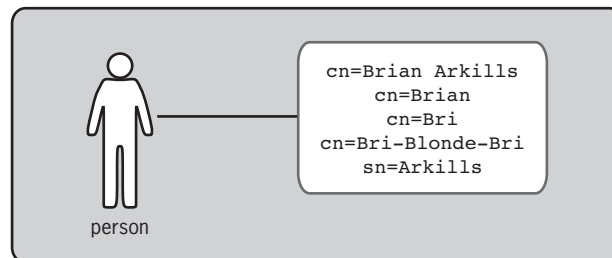


Figure 1-2 A person entry with a multivalued attribute

What's the Difference Between Object Class and Objectclass?

The terms are closely connected and very similar. `objectclass` is an attribute of an entry; you use the term only when you refer to a specific entry. Every entry in a directory has the `objectclass` attribute with one or more values that denote the object classes to which the entry belongs. An object class is a definition of rules that an entry of that object class will follow. The term "object class" is used to abstractly discuss a set of entries that follow the same rules. All of these entries have the same `objectclass` attribute value. It isn't the entries themselves that are being referenced; it is the set of rules that define that class of entries.

with a name, UPN serial number, and manufacturer attributes. You could delineate these different types of entries by using different object classes, or you could set up the entries to share the same object class, depending on the class's flexibility. The example shown in Figure 1-3 uses different object classes. Despite what is shown in Figure 1-3, different types of entries can exist side by side as long as structural rules don't prohibit such juxtaposition.

There is a special type of entry known as a container. A *container* helps to organize other entries by establishing a parent/child relationship. A commonly used container object class is *ou*, organizational unit. In my directory, we might want to place all of the person entries in a container named People while placing all the product entries in a container named Products. In general, this choice might make it easier to find or manage entries; however, in our example, you could just as easily find all the product entries by searching the directory for entries with the `objectclass` attribute equal to "part". But separating all the product and person entries into different containers makes it easy to delegate management of the entries to different people. For example, I might delegate management of the People OU to the HR department and of the Products OU to the product manager.

Container entries provide a structure for organization and management

10 Overview of LDAP

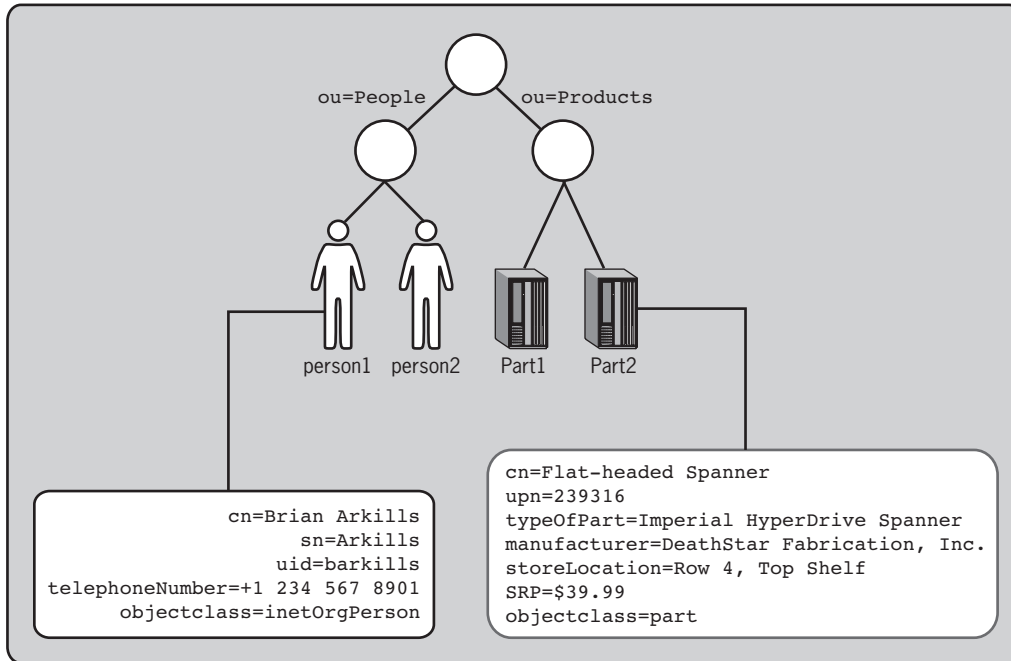


Figure 1-3 Person and product entries in separate directory containers

Containers have rules, but you decide how to use them to organize

Containers can have other containers as children, but child entries can have only a single container as a parent. So a pyramid (or upside-down tree) organizational structure is possible, but web or hub structures are not possible. To what extent containers are used is left to the details of implementation. You could choose to have an extensive structure of containers to provide critical organization, an arbitrary structure of containers, or no containers at all.

Content and Usefulness

The leading alternative to a directory is a database, and comparing the two helps illustrate both the nature of a directory and its usefulness. After this comparison, this section provides some examples to highlight typical directory content. Note that the

comparison considers only relational databases. Object databases are similar to directories, but the technology is not widely adopted.

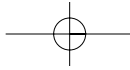
Directories Versus Databases

A directory usually contains entries that are static or change infrequently, because it is designed to provide very fast response to searches and lookups. A database often contains entries that can change frequently. Databases are designed to provide data that can be easily manipulated and sustain intense processing, with both reading and writing of that data. So if you want to keep track of your company's sales, you'd pick a database, not a directory, because (one hopes that) your company would be constantly writing new sales quotes. In contrast, your company's sales contacts would be best suited for a directory, because this information doesn't change often. In general, if the entries you'd like to store change less than once a day, a directory is probably the best solution.

*Directory=read;
database=read
and write*

Relational databases and directories also differ in terms of internal structure, and looking at this difference provides another measure of both what a directory is as well as how it is useful. Entries in databases have certain attributes that are called keys. Keys provide critical functionality for database technology by allowing you to sort entries. Additionally, you can use the keys to cross-reference information about an entry in one table to information about an entry in another table. The entries in a database have no inherent structural relationship to one another, and they don't really have names, aside from the keys (special attributes) that must be unique among all records in that table. In contrast, a directory is extremely structured. Each directory entry has a name that also defines that entry's location in a hierarchy. This relationship is discussed in more detail shortly. Another structural difference is that the attributes of a directory entry regularly have multiple values, whereas only denormalized relational databases have multiple values per attribute.

*Relational databases and
directories have
key structural
differences*



12 Overview of LDAP

The structural differences highlight special uses of databases and directories

These key structural differences mean that each technology has strengths and weaknesses that predispose it toward specific uses. Databases excel at storing objects that can be sorted in different ways. Databases usually implement a locking mechanism to prevent two parties from writing the same information, whereas directories don't. Complex queries that cross-reference multiple entries are typically quicker in a database than in a directory. Databases manage large data objects pretty well, whereas a directory is not designed for this purpose. The structure of a database lends itself to tables, whereas a directory is not well suited to store tables. Databases let you store procedures for efficient processing of complex requests.

Directories are suited for several commonly required purposes

Directories are really specialized data storage systems. Directories are much more suited for objects that need a hierarchy. Directories can be replicated across servers to allow access from multiple locations. They are more than a name service, because they allow both searching and retrieval, whereas name services just perform retrieval. Text-based information is particularly well suited for a directory because it can be easily searched; however, any type of data can be stored in a directory. Directories manage user attributes and policies well, because most services simply need to search and retrieve these

Do I Need to Choose Between a Database and a Directory?

No. They simply have different strengths and weaknesses. Each has a valid place, and it is likely that you will have both databases and directories. In fact, directories usually have a specially configured database running behind them. It might help you to think of a directory as a layer on top of a database, except that you can't access the database directly through normal means. In some cases, your company may want to synchronize some of the data elements stored in a directory and a database. For an example, see Appendix C.

attributes. Directories also manage information for machines and applications well, especially when the information is configuration-centered or is management information. Directories usually support a very fine level of access control, allowing information to be restricted as desired.

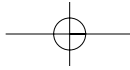
Typical Directory Use

You can use a directory to organize or manage just about any kind of information so people can easily find that information. Directories are most commonly used for personal information, but they can be used just as readily for information about any real-world object. For example, you could have a directory with products from your place of business. People could search the directory, based on the part number or type of product, to find information about the product they need and its physical location in a store. The directory could include pictures of the products and have a nice application interface (maybe Web-based) integrated with other functionality, such as an online ordering system, so products could be ordered.

You can use directories for information about people or real-world objects

The directory excels at storing personal information because information about people is fairly constant. Again, the directory is optimized to respond to queries about information that remains constant over time. Person-related information is of high value to the clients of a directory, whether they are people or applications. Person-related information also has a great need to be centrally managed so it is consistent, up-to-date, and secure. Think about personal contact information. The list is lengthy: a postal address, home address, office address, multiple phone numbers, e-mail addresses, a URL to a homepage, and so on. Obviously, there is more personal information than just contact information. However, personal contact information illustrates one inherent problem that a directory helps solve, in that I must first be in contact with you in order to get your contact information. A directory can let you store your personal contact information for easy retrieval by others—subject, of

Important personal information abounds, and a directory excels at storing it



14 Overview of LDAP

course, to the access controls applied to protect that contact information from untrustworthy folks who might use it to spam or harass you.

Applications use the directory on our behalf

However, the directory isn't just useful for others to find out about us. Often there are computer applications that need to check information associated with us. These applications do work for us behind the scenes. For example, it's fairly common for an e-mail service to query a directory with your e-mail address to find out which server your mailbox resides on, so it can deliver your e-mail to you. Additionally, some e-mail services automatically create an address book (which is stored in a central directory) so the user can simply pick a name instead of remembering an e-mail address. Many other applications and services are capable of looking up information in a directory, and some even provide an interface so people can modify directory information.

Authentication credentials can be placed in a directory

As another example, almost every time you log in, you are *authenticating* to some form of a directory. The directory validates the *credentials* you provide (a password or a ticket encrypted with your password) so everyone else on the network knows for certain that you are who you say you are. This authentication is critically important because it prevents someone else from impersonating you. Many network operating systems (NOSs) use LDAP as the basis of their internal directory functionality. This close integration can be an incredible benefit but also can have some drawbacks. I'll look a bit more at how a NOS might use LDAP shortly.

People are more productive with a directory to support them

Without the directory behind the example situations I've explored, people would be much less productive. Managing e-mail addresses and authenticating to every network resource (instead of authenticating just once) are tasks that people don't want to be bothered with, and the directory helps people manage this information. Directories have many of these behind-the-scenes uses, which ultimately benefit all of us. In fact, the

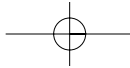
major benefits of a directory are behind-the-scenes types of services, with a computer application or a computer running more smoothly because the directory is there.

Another perfect use for a directory is in managing machines. Networked machines inevitably have configurations that need to be managed. These configurations are largely static, but keeping the information centrally lets changes be easily implemented. Networked computers have even more specialized uses for a directory. Computers that are members of a NOS service usually need to authenticate themselves. This information needs to be centrally maintained. Computers also have many characteristics such as software, environment configuration, and access privileges. Network administrators appreciate any tool that will help them manage this information, which generally changes infrequently. Microsoft's Active Directory (an extension of Windows 2000 Server) provides a good example of a specific implementation of this type of use. For example, Active Directory allows a directory administrator to define *group policy* directory entries that are a set of configuration information, and apply these policy entries to computer entries. This process facilitates computer management and demonstrates one way in which LDAP can be used.

Machine and computer management information belong in a directory

Users also experience the benefit of machine management via a directory. The previous examples may appear to benefit only computers and computer support personnel, but they also benefit other employees and customers of the company. Users don't want to memorize obscure naming conventions in order to find a network resource. Directories can help address this issue by helping users locate network resources via the directory. Imagine, if you will, the harried user who desperately needs to print a document for an important presentation in a remote location at your place of business. How does she find a printer? There are no support personnel at hand. The directory comes to the rescue, because it knows where all the printers are and the user can easily ask the directory for a printer at that

Directory management of computers helps people



16 Overview of LDAP

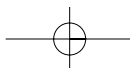
location. The directory might interface with the user's laptop to configure the needed printer settings. The directory might further address the issue of multiple obscure naming conventions by providing a unified and user-friendly naming convention that hides the real naming conventions being used.

Benefits of a Directory

Many IT personnel know that implementing a directory is important for their business but don't quite know how to justify the cost and effort required to their managers. Benefits 1-1 consolidates the relevant points into a useful form that you can use in such a situation. Benefits 1-2 later in the chapter is a similar summary that focuses on the benefits unique to LDAP. Use Benefits 1-1, Benefits 1-2, and Figure 1-4 to begin to build a case to your manager for implementing an LDAP directory.

Benefits 1-1 Benefits of a directory

- Make network administration easier
 - Central management of people information
 - Central management of computer and machine configuration
 - Central management of user accounts
 - Reduced support costs from centralized management
- Unify access to network resources
 - Uniform naming convention
 - Potential for single login to network resources
- Provide single destination for users to search for information
 - Contact information
 - Central location of network resources
 - Potential as a catalog for any data, for example, product documentation



- Improve data management
 - Improve the consistency of data that is widely used
 - Provide centrally managed security for business-critical data
 - Organize data in a logical structure
- Help streamline business processes
- Provide repository and lookup for application and service data

Introducing LDAP

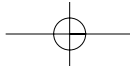
How does LDAP work? Why has LDAP been adopted as the directory standard by so many large companies, as well as by all the major software vendors? This section provides an overview of how LDAP technology functions and why LDAP is considered so highly by the industry. Later chapters in the book expound on this overview in more detail.

LDAP (Lightweight Directory Access Protocol) originated out of the X.500 series of International Telecommunication Union (ITU) recommendations. ITU is an international standards body, and X.500 is a set of recommendations about directories. Because of this relationship, the structure of X.500 and LDAP directories is similar. LDAP directory implementations are often also X.500 compliant, and gateways between the two directories are also plentiful. LDAP was pioneered at the University of Michigan, and there is still a free implementation available from their Web site, along with documentation, source code, and other resources.

LDAP is defined by a set of published Internet standards, commonly referenced by their Request For Comment (RFC) number as published at the IETF Web site: <http://www.ietf.org>. The Internet Engineering Task Force (IETF) helps manage a rigorous

LDAP came from X.500

A set of nine RFC documents defines LDAP



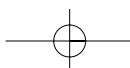
What Happened to X.500?

There were a host of problems with X.500. It was too tied to the OSI (Open Systems Interconnection) protocols, and so wasn't well suited for the TCP-dominated world that emerged. It used a complicated encoding mechanism (although to be fair, LDAP uses pretty much the same one). Its creators were very ambitious, and so X.500 was probably too complicated for the kinds of problems that people really wanted to solve. And finally, X.500 was meant to be a global directory service, even though it wasn't clear that everyone thought this was a good problem to solve. In short, LDAP and the Domain Naming System (DNS) solved in a simpler fashion the real problems that people faced.

proposal process in which ideas such as LDAP are reviewed in drafts until they are ready to be published as an Internet standard. Don't be confused about the number of RFC documents associated with LDAP. LDAP version 3 (v3) is defined by nine RFC documents. RFC's 2251 through 2256 give the core details, and were later followed by RFC 2829 and 2830. RFC 3377 followed shortly prior to the time this book went to press. It tied all of these RFCs together as the official LDAP v3 standard. In addition to these nine documents, you will find many other documents that address technology based on the core LDAP standard. LDAP comprises a wide set of technology and continues to be developed, so several documents are needed to help define its many facets. This book includes coverage of the material in the core RFCs as well as most of the other LDAP RFCs. Although the RFC documents define the standards, they don't tell the whole story, and they are certainly not enjoyable reading. But for further reference, when the RFC documents provide more detail than is appropriate for this book, they will be cited.

To summarize LDAP, we'll be looking at four areas:

- Namespace
- Client operations



- Schema
- Management

These four areas coincide with the next four chapters of the book, which expand the summary information found here in greater detail. In the second section of the book, each of the chapters looks at specific vendor implementations. These chapters also use these four primary areas to organize the information.

Mycompany.com

Prior to looking at the four LDAP areas, I need to introduce the example company that is used throughout the book to provide a concrete context for abstract concepts. Mycompany.com is a typical company, with sophisticated technical requirements for carrying out its business. I've intentionally left the profile of Mycompany generic, to maximize the relevance of the example. Figure 1-4 shows a representative sample of the types of business applications and IT infrastructure services that Mycompany has deployed and would like to integrate with an LDAP directory.

Trying to Read the RFCs?

If you try to read the RFCs on the IETF Web site, you may encounter several problems. You may come across references to X.500 documents that you can't find online. This is because X.500 is maintained by the ITU international standards body. ITU asks that you pay to receive a copy of its standards, and you can order a copy online from its Web site. Alternatively, I've listed a few online X.500 references at the back of the book, including an entire online book on X.500. Second, you may not understand the special coding system used in some of the definitions. It is called Backus-Naur Form (BNF), and you can read more about it in RFC 822. The RFC is oriented toward simplifying the encoding of e-mail; but if you skip several of the messaging-specific parts, you can get an idea of how to use the BNF format.

20 Overview of LDAP

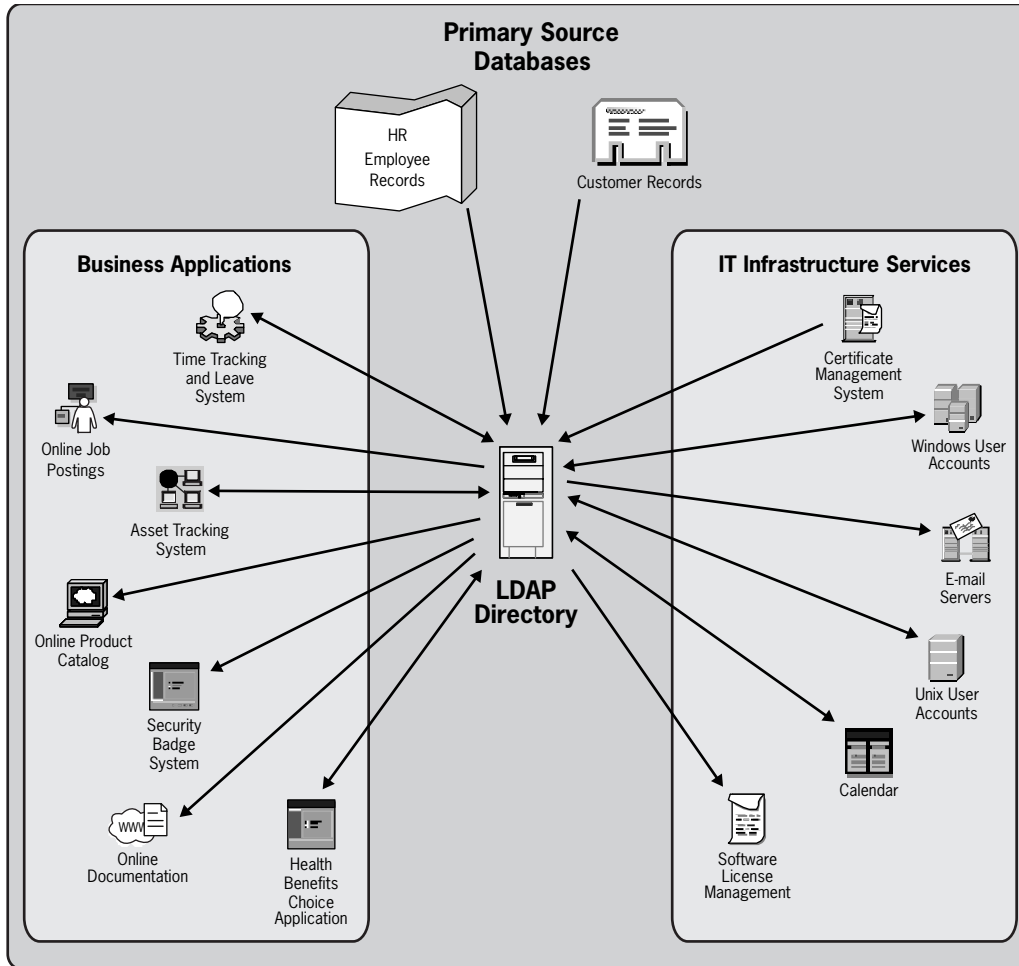


Figure 1-4 Integration of Mycompany.com's applications and infrastructure with LDAP

Mycompany would like to use an LDAP directory to tie these applications and services together to simplify data management, cut development and support costs, and provide a single point of IT infrastructure management. Figure 1-4 further shows how each of these applications, databases, and services might interact with data in the LDAP directory. Arrows out of the directory represent a search operation (also called a query) that is the

source of information for the service or application. Arrows into the directory represent a source of directory data or potential modification of existing directory data.

Namespace

Every directory needs a namespace. What, you might ask, is a namespace? As you might expect, *namespace* refers primarily to how entries are named. However, it can also imply other things, such as an organizational structure for the entries. Incidentally, the term “namespace” can also be used in a general sense to refer to all the objects in a specific container.

In general, the LDAP namespace is the system used to reference objects in an LDAP directory. Each object must have a name, and the name of each object serves two purposes. First, it allows the object to be referenced. Second, it allows the object to be organized into a logical structure. Understanding the namespace is the key to understanding the structure of the directory.

Each entry in the directory needs a name for the purposes of referencing that entry. These names must be unique in an LDAP directory so you can designate a specific entry. But instead of simply naming each entry with a unique name, the namespace goes a step further and designates where in the directory’s organizational structure each entry belongs. So if you know the name of an entry, you also know where that entry resides in the directory structure.

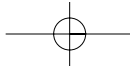
Because the namespace is organized in a hierarchical fashion, management control can be delegated at multiple points in the hierarchical structure. The hierarchy that is inherent in the namespace conveniently provides an effective means for cooperative delegation of management. This is a significant advantage of LDAP over databases, and it is usually one of the primary factors in deciding how to organize data in the directory.

To find information in a directory, a common set of naming rules is needed; these rules are called a namespace

The namespace serves two functions: to identify objects and to define the hierarchical structure

Because each entry indicates a location in the directory, its name must be unique

Namespace hierarchy allows management control



22 Overview of LDAP

DNS is one common namespace

Many of the directories you may have used share a common namespace, which happens to be an Internet standard: DNS. For example, when you send an e-mail to another person's mailbox across the Internet, you address it in a way (person@domain.com) that conforms to the DNS namespace. The e-mail is delivered to only one person because the mail service using the DNS namespace also enforces uniqueness of names. DNS provides a namespace for many computer services.

The LDAP namespace is very similar to DNS, and DNS can be employed

DNS is by definition hierarchical in nature. The LDAP namespace is hierarchical too. Because the namespaces are so similar, many LDAP directories leverage the DNS namespace, so the LDAP namespace works seamlessly with DNS. This reliance helps make LDAP more attractive and provides for future development of globally integrated LDAP directories. LDAP vendors that adopt DNS compatibility allow for the possibility of seemingly independent directories being more easily connected in a global hierarchy at a later time, just as an intranet-based DNS zone might be connected to the global DNS namespace. Chapter 5 examines the integration of independent directories, and Chapter 2 introduces some of the primary concepts. Formalizing the relationship of LDAP to DNS is one of the tasks of an IETF working group; Chapter 2 also examines this relationship.

DNS is not required but usually is preferred

With some directory servers, clients or users can automatically locate directory servers for their local DNS zone without any prior knowledge or configuration (for more detail, see Chapter 2). But to be clear, LDAP does not require that DNS be used in forming a directory namespace. With the help of a name resolution service like DNS, a client locates a directory server on the network. Other name resolution services can be used to locate the LDAP server; however, the trend is definitely to implement LDAP with a reliance on the DNS namespace. The benefits of doing so are greater than the alternative, but there are reasons not to do so as well. These reasons are usually limited to LDAP directories with an isolated use.

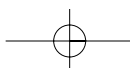


Figure 1-5 shows a simple version of Mycompany's directory. The name of the root of the directory is known as the directory's *base DN*. The directory root is not necessarily a directory entry. The server's base DN typically matches the DNS name of the directory server and uses the *domain components (dc)* attribute to represent the DNS zones. However, the server's base DN does not necessarily have to coincide with the server's DNS name. The directory server's base DN might be different to allow greater flexibility in designing a distributed directory architecture across multiple directory servers. The flexibility to create a distributed directory via the namespace is a key advantage of LDAP over databases. Chapter 2 covers some of the foundational concepts behind a distributed directory architecture, like referrals, replication, and the full details of namespace. Chapter 5 addresses distributed directory architecture models, as well as the issues and solutions to integrating directories.

The root of the directory has a name

But how do you reference an entry within my LDAP directory? Each entry in the directory has a unique name known as the *distinguished name (DN)*. Each entry also has a name local to its immediate container known as the *relative distinguished name (RDN)*. The RDN is unique among all entries in that container. For now, think of a container as being similar to a directory or folder in a file system; Chapter 2 covers containers in more detail. The DN of each entry is formed by concatenating the RDN of the entry with the RDN of the containers between the entry and the directory root. There is a comma between the RDNs in the DN. Neither the DN nor the RDN is an attribute of the entry, but the RDN consists of one of the attributes of the entry.

The DN is the name of an entry

The RDN is an attribute type and value pair. More precisely, it can be any attribute pair (or combination of attributes) that is unique in the entry's immediate directory container. The RDN does not have to be unique across the entire directory. You can compare the RDN to the hostname, like myserver, which is

The RDN is the local name of an entry

24 Overview of LDAP

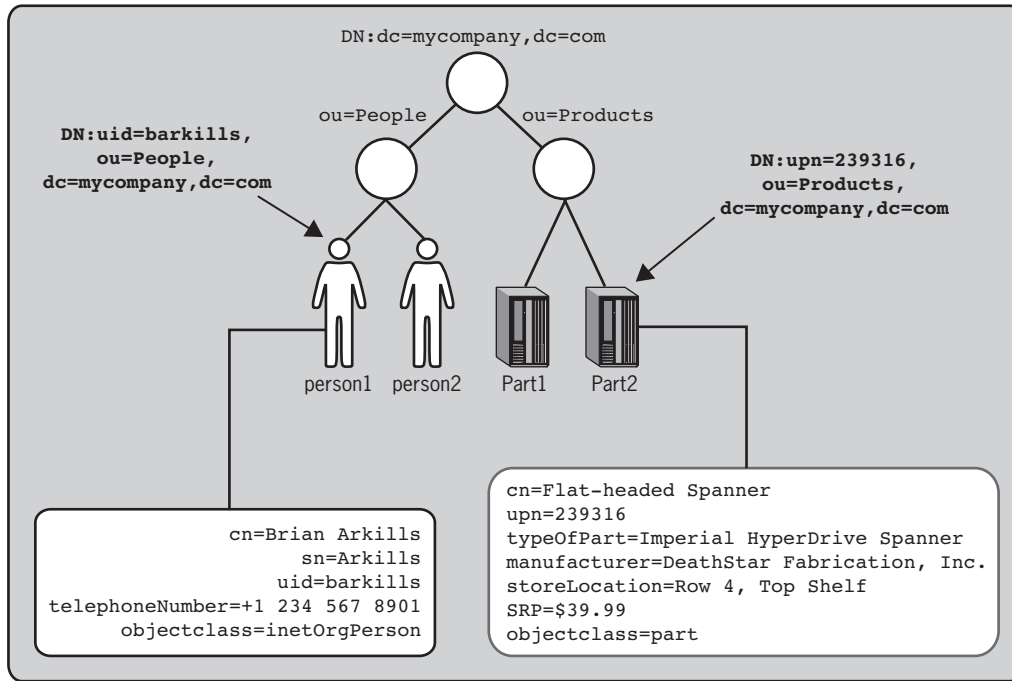
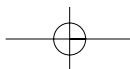


Figure 1-5 Person and part records with DN in an LDAP directory that integrates with the DNS namespace

unique within the mycompany.com DNS zone, but is not necessarily unique among all DNS zones in the world.

Example to illustrate DN usage

The DN of the person entry shown in Figure 1-5 could have been `cn=Brian Arkills,ou=People,dc=mycompany,dc=com` instead of `uid=barkills,ou=People,dc=mycompany,dc=com`. Notice that each RDN component includes both the attribute type and value. For example, the single component `cn=Brian Arkills` has both the attribute type `cn` and the attribute value `Brian Arkills`. The attribute value without the attribute type would not be sufficient to distinguish the entry, because the value might refer to different attribute types on many entries. The *common name* (`cn`) of the entry in Figure 1-5 is Brian Arkills. Note that `cn=Brian Arkills`

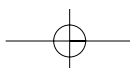


must be unique among all entries in the container `ou=People` to qualify as an RDN. As you might realize, uniqueness of a person's name isn't guaranteed, so another attribute is often used instead as the RDN. Mycompany might choose the user identity `uid=barkills` instead as the entry's RDN, because login IDs are unique within Mycompany. The DN `uid=barkills, ou=People, dc=mycompany, dc=com` refers to exactly the same record in the directory namespace as the DN above. This second DN simply uses a different RDN to identify the entry desired, where the *uid* is my user identity or account name.

Based on the desired integration noted in Figure 1-4, Mycompany's directory namespace might look something like Figure 1-6. For simplicity, almost no directory entries have been shown, but each of the containers shown (open circles) would have entries (closed circles) and possibly additional containers for further organization or delegation. For example, the People namespace might be divided with containers by department, with the entire Sales department in a container and the entire Engineering department in another. The layout shown in Figure 1-6 could be implemented differently and still meet the desired integration requirements.

The namespace that LDAP employs has substantial benefits. First, it provides a naming model that uniquely identifies entries but is flexible in that more than one name may be valid. Second, it is inherently hierarchical. This allows entries with the same naming attribute to exist in the directory in different containers. It provides a vehicle for delegation of management, application of access controls, and organization of data. Third, it usually leverages DNS, which gives an LDAP directory an advantage in integrating with other technologies, and service location resolution from anywhere. Fourth, the namespace allows LDAP to distribute a directory across multiple servers. For more detail on this topic, see Chapter 5. This benefit is significant because greater reliability, distributed load, and localized

LDAP's namespace provides many advantages



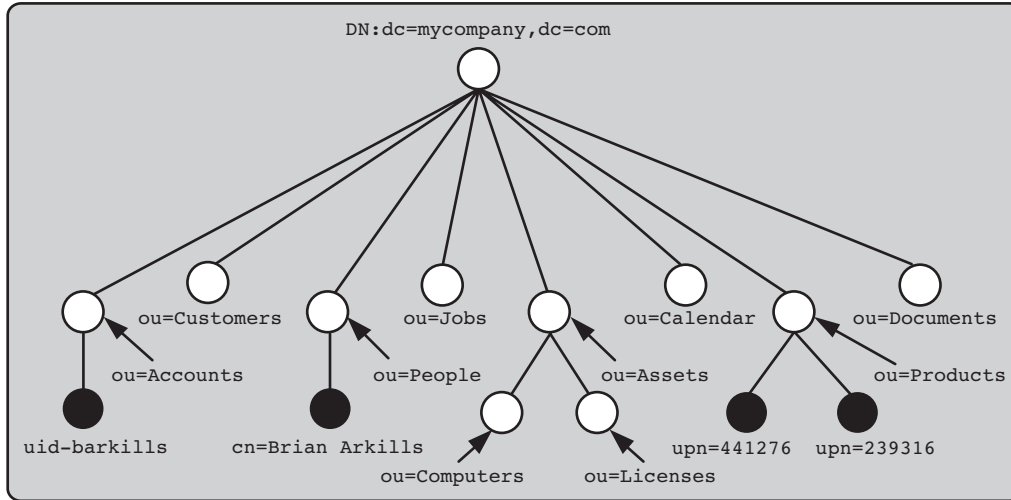


Figure 1-6 Mycompany directory namespace

directory data are additional benefits that can be realized from this distribution.

Protocol

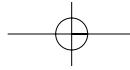
LDAP is primarily a set of server operations

At its heart, LDAP defines a set of server operations (the directory access protocols) used to manipulate the data stored by the directory. But there are many other aspects to LDAP, and people are hard at work developing draft standards that might be added to the accepted Internet standards that comprise LDAP. Some of these extensions may comprise the rules that govern the method in which data is stored in an LDAP directory, extensions of the protocol and server operations, standards for secure client authorization, architecture for ensuring directory reliability, and so forth.

Client-Server Model

TCP/IP is required for LDAP

As an Internet protocol, LDAP uses TCP/IP for its communications. For a client to be able to connect to an LDAP directory, it must open a TCP/IP session with the LDAP server. LDAP mini-



mizes the overhead to establish a session allowing multiple operations from the same client session. It also gains traffic efficiencies from compression because the majority of data stored in the directory is discrete text-based information. LDAP employs *BER encoding* to encode the attribute value data passed between the server and client. This overly complicated encoding method is retained from LDAP's X.500 roots.

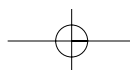
Interestingly enough, the set of LDAP operations correspond one to one to a set of standard application programming interfaces (APIs) in different languages. An *API* is a set of functions that programmers can use in writing software. These functions provide a higher-level deliverable by hiding the messy guts of the code from those that use it. Some APIs are *closed source*, meaning that the guts are intentionally hidden from everyone. Others are *open source*, meaning that anyone can view the details and even contribute improvements. The LDAP APIs are all open source. For an example of a function from an LDAP API, take the server operation used to add entries. There is a standard LDAP function `ldap_add()` in the C language API that an application would use to ask the server to perform the add operation. Similarly, the standard APIs define functions for each of the server operations defined by the LDAP specifications. Any application that used an LDAP API to interact with an LDAP server is called *LDAP-enabled*. The standard C language version of the API is documented in RFC 1823.

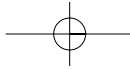
Any LDAP client can speak with any LDAP server

Clients

The LDAP client can be either standalone software that a person interacts with by typing in the syntax as required, or it can be an integrated piece of software with much of the operation automated and the syntax requirements hidden from the user. For example, the Windows 2000 operating system has integrated the LDAP client functionality into several of its core applications. In Windows 2000, you can choose the Search option from the Start menu, and look up person or printer entries in the Microsoft Active Directory (in other words, the

The LDAP client can be standalone or integrated software





28 Overview of LDAP

LDAP server) to which the computer is connected. There are many LDAP-enabled Web sites that provide a single interface (sometimes called portals) for people to use in searching and modifying entries in their company's LDAP server. In addition to these Web sites, most modern browsers support the LDAP protocol and are completely functional clients for retrieving information from LDAP. The flexibility of integration is one of the primary reasons why many software companies have embraced the LDAP protocol.

The open standards model means LDAP is very easy to integrate

The beauty of the LDAP open standard becomes evident when you realize that any LDAP client or LDAP-enabled application can successfully communicate with any LDAP server, regardless of the client's or server's particular operating system. The open standard, multiple-platform model makes integration easy, in a marketplace that makes integration difficult. This means that implementing LDAP in a complex, nonhomogeneous operating system environment is significantly easier than implementing other technology.

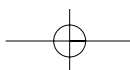
LDAP has only ten operations, and this is good

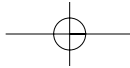
Operations

There are ten LDAP operations. The limited number of operations is quite important, in that client programs that interact with the directory are much simpler than client programs that interact with other similar technologies. These operations can be grouped into three basic categories with one outlier, as shown in Table 1-1.

Table 1-1 LDAP operations

Category	LDAP Operations
Client Session Operations	Bind, unbind, and abandon
Query and Retrieval Operations	Search and compare
Modification Operations	Add, modify, modifyRDN, and delete
Extended	Extended





The extended operation is unique among the operations. The extended operation is a placeholder for specific directory implementations to extend the functionality of the protocol but still have a predefined syntax for doing so. The LDAP designers showed a lot of forethought by including the extended operation. By standardizing a way to expand the operational functionality, they eliminated any perceived liability in the limited number of operations.

The client session operations help to control the client-server session context for all subsequent LDAP operation requests from that client. The bind and unbind operations allow the client to establish an identity with the directory. This identity can be used by the directory to determine authorization to perform the other operations, and can be used to control access to directory information. The abandon operation allows the client to cancel an outstanding operation request.

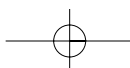
Session operations affect how the client interacts with the directory

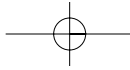
The query operations allow the client to look up information in the directory. Most readers new to LDAP need to know how to intelligently search an LDAP directory. The search operation is the most frequently used, and skill in using it will be repeatedly valuable. The search operation has many parameters, in fact, more parameters than any other operation. This greater complexity is worthwhile, however, because it allows the user to designate sophisticated queries in a search for data within the directory. Because of the importance of the search operation, be sure to closely peruse the details and examples provided in Chapter 3. The compare operation allows a client to request a verification of the information associated with an entry. The client sends the purported value(s) of the entry, and the server responds with success if it matches or failure if it doesn't.

Query operations allow data to be found and retrieved

The modification operations allow the client to change information in the directory. These operations might be restricted in some instances, for example, in the case of a public read-only directory. Of these operations, the modifyRDN operation is the

Modification operations allow a variety of changes to be made





only one in need of any summary explanation. The modifyRDN operation allows the client to change the name of an entry and possibly move the entry to a different container.

Referrals and Unicode support extend the functionality of LDAP

Two other notable protocol features of LDAP are referrals and Unicode UTF-8 support. *Referrals* allow an LDAP server to redirect a client to a different LDAP server to locate the data the client requests. This functionality enables integration or cooperation between directory servers and even independent directories. *Unicode* is a specific method of representing data in character sets that are specific to a language or locale. This means that any written language can be represented if Unicode is used to encode the data. Not all data is represented in Unicode; for example, you may have heard of ASCII encoding. Data encoded in ASCII would not allow most of the languages in the world to be represented. Unicode support extends the usefulness of LDAP to virtually any language, making LDAP a global solution. For details of the LDAP protocol, see Chapter 3.

Schema

The schema defines the rules

A game without rules is chaotic and subject to the players' whims. The popular comic strip *Calvin and Hobbes's* "Calvinball" game zealously demonstrates how out of control life can be without rules. In this game, Calvin and Hobbes make up the rules as they play, which inevitably leads to disaster. The set of rules that defines what types of entries can be in the

Is LDAP the Final Word for Directories?

Probably not. In fact, there is evidence that the Web services movement with the XML standards may add something significant to the future of directories. A new standard called DSML is emerging. This standard, however, assumes the presence of an LDAP directory. Perhaps in the future it will evolve beyond this. For more information on DSML, see Chapter 5.

directory is known as the *schema*. If a particular object class isn't in the schema, you can't create an entry with that object class. You extend the schema to include a new object class or to allow new optional attributes on an existing object class. The schema further defines pertinent rules like what type of value can be placed in an attribute, and what operators are valid for those attributes. The *operators* are what the directory uses to compare one attribute's data value to another value. Greater than, less than, and equality are examples of common data operators.

Schema Checking

The addition of any new entry in your directory is subject to a schema-checking process. Should any of the data not meet the applicable definitions, the addition of the entire entry fails. The schema is not something one can ignore—it has teeth that bite! Some LDAP implementations allow you to turn off this schema checking, but doing so is not wise. The data would lose its uniformity.

All new entries must pass the schema-checking process

Default Schema

The minimum set of schema objects required by the LDAP standard, as listed in RFC 2252 and 2256, will give Mycompany a functional directory. The minimum LDAP schema is largely formed from the set of X.500-defined schema objects and follows the basic rules for the X.500 schema. This is the key reason why so many LDAP products can also be X.500 compliant. Directory vendors take care of implementing this minimum set of schema objects, so you only need to be familiar with what these schema objects are and how you might use them. Most software vendors that leverage a directory find this minimum set insufficient for their purposes and further extend the schema with their own definitions.

The LDAP schema comes from X.500

Extending the Schema

Although the schema is arcane because of its syntax format, it is also the source of most of the flexibility of LDAP. Mycompany's directory can implement schema extensions to include whatever

Schema syntax is hard to read, but it is very flexible

types of data the company deems necessary. The schema also lets you define new ways to interact with the directory and new ways to work with the data. Chapter 4 considers schema extensions that others have found significant.

The schema is published so clients know what is supported

LDAP publishes the directory schema so any client can determine what definitions and rules the server employs. The location where the schema is published is stored on every entry, and this information tells you where to look for the schema. The location is called the *root DSE* (Directory Systems Agent Specific Entry) container. Most LDAP directories have a single schema that applies to the whole directory, so the location is the same for all entries. Some LDAP servers may allow the definition of unique schemas for different parts of the directory.

Here is a sample schema definition for the `person` object class:

```
person OBJECT-CLASS ::= {
    SUBCLASS OF { top }
    KIND abstract
    MUST CONTAIN { sn, | cn}
    MAY CONTAIN { userPassword | telephoneNumber |
        seeAlso | description }
    ID 2.5.6.6}
```

If your curiosity is piqued, see Chapter 4, which addresses the schema in more detail.

Management

Information that is centrally organized in an LDAP directory lends itself to management. In fact, an LDAP directory can become the hub of IT management. Figure 1-4 shows how centrally important the LDAP directory is to Mycompany.com, and you can easily imagine how this architecture would make management tasks easier. The support for LDAP directory management functionality is therefore very important. Management functionality that is easy to use or provides ways to simplify integration is highly desirable. Some directory management

Does the Schema Look Complex and Tedious?

For the most part, it is both complex and tedious. Don't underestimate the importance of the schema. You may feel like the schema is boring, and the syntax not worth learning. However, the schema employed by LDAP is one of the greatest features of LDAP, because it lets you design how data is represented, what data is allowed, where it is allowed, and what additional operations can be performed on that data. The other areas of LDAP may be flashy, but they all derive their functionality from definitions in the schema. Without the flexible model LDAP uses with the schema, a lot of the flashiness and all of the extensibility would be gone.

functionality is not addressed by the LDAP standards. For example, LDAP leaves the implementation of storing and retrieving the data on the server up to the vendor. Usually, a specialized database is used. This functionality can introduce a variety of management tasks and functionality, depending on what software component is chosen. In most cases, management tools are best left to vendors and market-driven competition.

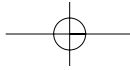
Distributed Directory

Possessing a fault-tolerant directory is of high importance to Mycompany because the company wants to make the directory a central focus of critical business data and processes. Most vendors have implemented some form of *replication* to allow portions of the directory to be copied to multiple servers. Employing replication copies a directory or portions of a directory across multiple LDAP servers. Distribution via replication provides several advantages, including load distribution and protection against data loss. For more details on replication, see Chapter 5.

However, replication is not the only way to have a distributed directory. Different LDAP servers can host different portions of the directory namespace with references to the other LDAP

The LDAP standard leaves the implementation of replication to vendors

A distributed directory has many advantages



servers. The namespace can be divided in any way desired. This type of distribution provides advantages, such as storing information about your European office on LDAP servers located in that office. This would decrease the dependency on long-distance connectivity between geographical locations. Another advantage might allow politically divergent departments to run their own server. This would enable separate management of information that each department feels that it owns. The flexibility that LDAP provides in terms of namespace provides benefits in simplified management.

Integration and Data Manipulation

Integrating independent LDAP directories and servers from rogue departments or mergers will further empower the value of Mycompany's directory. As discussed previously, LDAP supports referrals that allow one LDAP server to reference another. These referrals can be used to connect servers. Integration via other methods can also be critical to keeping data consistent and authoritatively controlled across your organization. For some other methods, see Chapter 5.

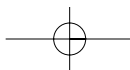
LDIF and other features allow duplication of data between servers

There is also an LDAP standard that allows directory data to be copied between servers. The *LDAP Data Interchange Format (LDIF)* standard provides a means for a directory administrator to move directory data between servers in a file format. The LDIF standard also gives the administrator a way to make batch modifications to many entries using search-and-replace text-manipulation tools. For more detail on the solutions LDAP provides in this area, see Chapter 5.

Authentication, authorization, and encryption are needed to secure information

Security

The term "security" is used in a broad sense in the computer industry. In discussions of computer security, typically two areas are of concern: authentication and authorization. *Authentication* is the means of proving we are who we say we are. *Authorization* is the means of designating access permissions to users. When we add the complexity of operating on a



network, there is a third area of concern: privacy. *Privacy* is the means of ensuring that data is kept safe so that it is available only to those for whom it is intended. Some form of encryption is usually used to keep data private.

LDAP supports many authentication methods. LDAP v2 and v3 both support *simple authentication* (cleartext), Kerberos, and *digital certificates*. LDAP v2 supports Kerberos v4, whereas LDAP v3 adds support for Kerberos v5 via *Simple Authentication and Security Layer* (SASL). You can find the standard designation for Kerberos under RFC 1510 and 1964. LDAP v3 also supports the SASL. SASL offers a way to add authentication mechanisms to any protocol. Examples of such mechanisms are Kerberos 5 and DIGEST-MD5. The SASL standard designation is specified in RFC 2222. LDAP v3 identifies Transport Layer Security (TLS), the successor to Secure Sockets Layer (SSL), as the way to authenticate with digital certificates. TLS is documented in RFC 2246. Given that LDAP is an Internet standard, you are probably not surprised to learn that Kerberos, SASL, and TLS are also Internet standards.

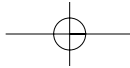
LDAP supports cleartext, Kerberos, and SASL for authentication

Currently, LDAP has no specifications for authorization. With authorization or access control, you can control access to entries and even attributes. You can allow or deny specific users or groups of users access to entries. The access levels vary between permission levels such as being able to read or write to the entry. Because there is no agreement on authorization in the LDAP standard, vendors are left to implement their own authorization model. Discussion and work is under way in the IETF on this subject, so in the future this may be part of the LDAP standard. This lack of standardization means that this is one of the areas Mycompany will want to scrutinize closely with respect to its choice of a vendor.

Implementation of authorization is left to the vendor

LDAP also supports session encryption for privacy. All information that is communicated during the client-server session can be encrypted so eavesdroppers are foiled. Both SSL and TLS are

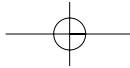
LDAP supports SSL and TLS for data encryption privacy



supported. TLS is the successor to SSL, and can be thought of as SSL v3.1. Both SSL and TLS are based on *public key certificate* technology, which depends on the server having a certificate from a trusted certificate authority server. Additionally, clients can obtain certificates and in some cases use them as a valid form of identification (authentication). Although this requirement may pose some hurdles, including management of certificates, the resulting security can be deemed well worth the trouble. Encrypting the session can provide privacy to the directory information passed over the wire. The information in your organization's directory may be some of the most sensitive data you have, possibly personal and proprietary, and your organization won't want this information passed "in the clear" for the advantage of outsiders. Session encryption is critically important if simple authentication (cleartext) is the only available authentication method for a specific client.

Does LDAP Provide Security?

I often talk to people who confuse LDAP as a generalized solution to security concerns. There is a perception that "LDAP" is a security buzzword. LDAP is a directory technology, not a security technology. LDAP directories commonly use security technology, like Kerberos, to provide secure access to the directory, and they may also be used to store security-related information such as X.509 certificates or authorization information. But LDAP itself doesn't provide any security services. LDAP as a standard or a protocol still has room to grow before I'd associate security with it. Don't get me wrong, in most products it is implemented relatively securely. However, the IETF should standardize many of the security features that vendors have implemented. Additionally, the protocol should require some input stream validation mechanism so a client can't pass commands to the directory server via a buffer overflow or other nastiness.



Vendor LDAP Products

Part II covers several of the most popular LDAP products. A few notes about the trends and diversity of offerings are worth mentioning here.

One strong trend has vendors using LDAP to support their network operating system (NOS). This is innovative and provides nice opportunities to integrate many infrastructure services with all the benefits of LDAP. However, the offerings from these companies usually limit the integration with products outside that vendor's software suite. This is made worse when you must employ the vendor's NOS to use the LDAP directory. Integration is nice, but not integration without freedom to choose the best components.

Vendors often use a directory to enable their network operating system

Another trend that has continued throughout LDAP's history is the open source movement. This movement is important in light of the previous trend. The open source movement has helped ensure that some minimum level of integration is kept standard and, in turn, has put pressure on vendors to work with others. Open source LDAP software offers the ability to choose components and eliminate dependence on a single vendor.

LDAP has a strong open source movement

A final trend to note is that almost every large software company has an LDAP directory offering. In addition, several small company offerings also offer LDAP directory products. This diversity of products is great for the consumer, because it provides greater choice and means that vendors have to provide real competitive advantages to capture our attention. Table 1-2 lists most of the LDAP server offerings.

There are a lot of LDAP product offerings available

Why Choose LDAP?

The reasons to use LDAP are overwhelming. Simply put, LDAP is the best show in town if you want to use a directory.

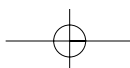


Table 1-2 LDAP servers

Vendor	Product Name
Computer Associates	eTrust Directory
Critical Path	CP Directory Server
IBM	SecureWay
Sun AND Netscape	Directory Server (used to be iPlanet and Netscape Directory Server)
Microsoft	Exchange 5.5 AND Active Directory
Netscape	Directory Server (no longer offered)
Novell	eDirectory (formerly NDS)
OpenLDAP	OpenLDAP
Oracle	Internet Directory
Syntegra	Global Directory AND Aphelion Directory
University of Michigan	Slapd

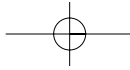
Consider the many companies and organizations that have already adopted the technology. Benefits 1-2 summarizes a list of advantages of LDAP, but you probably are already familiar with them. This list, together with Benefits 1-1 and Figure 1-4, would make a good start on justification for deploying a directory.

Benefits 1-2 Summary of LDAP Advantages

- Entries are organized in a distinct hierarchy. This provides the means to delegate administration, apply access controls, and enjoy other information management benefits. Even the name of an entry reveals information about the entry.
- Attributes of an entry can have more than one value. The structure of an entry doesn't need to be extended to per-

mit additional data. An entry can also have multiple names, each of which is unique across the directory.

- An LDAP directory can be distributed across multiple servers. This design distributes the load and provides other management benefits.
- LDAP is an open standard, with multiple-platform support. An LDAP client on any platform can communicate with any LDAP server. So there is less reliance on a single vendor.
- The LDAP client requires very few resources to run, and it can easily be integrated into other software. The LDAP operations are few in number, which makes it easy to interact with the directory. Session traffic is encoded and uses TCP, so network communications are economical.
- LDAP has a standardized API for multiple platforms. As a result, your developers can leverage the information in the directory when developing new applications instead of having to rebuild this information. This saves money and time, while opening up the possibility for new cross-functional applications based on access to data that was not previously available.
- LDAP provides easy integration with existing standards because LDAP uses other accepted standards, including TCP/IP and DNS. Standardization of integration methods with other standards is an ongoing process. Later chapters note the results of this process.
- LDAP supports strong authentication and encryption methods.
- LDAP uses Unicode UTF-8 so almost any language character set can be represented. This makes an LDAP directory capable of supporting international organizations in the native language.



40 Overview of LDAP

- LDAP employs an extensible schema, which allows further operability to be added. Operations and data must conform to the schema, which improves the quality of data.
- The usefulness of LDAP is being extended constantly, because of the widespread adoption of the view that it is the future of directory services.

Cross-technology integration has become a strength of LDAP

Most organizations are now taking the next step and looking for further ways to integrate LDAP with existing technologies. This step extends the usefulness and value of the investment made in LDAP. There are many examples of leveraging other technologies to extend the usefulness of LDAP. This is because LDAP is based on a clear standard that easily integrates with other existing standards.

