CHAPTER 3

Novell eDirectory

This chapter covers the following testing objectives for *Novell Course 3001: Foundations of Novell Networking*:

1. Identify basic Directory Service tasks.
2. Identify common Directory Service uses.
3. Describe how a Directory is structured.
4. Identify the role and benefits of eDirectory.
5. Identify how eDirectory 8.6 works.
6. Identify and describe the composition of eDirectory.
7. Identify and describe eDirectory object classes.
8. Identify the flow and design of the eDirectory tree.

NetWare 6 introduces eDirectory 8.6, the greatest version to date of Novell's world-class directory service.

eDirectory is the world's leading Directory service. It provides a unifying, cross-platform infrastructure for managing, securing, accessing, and developing all major components of your network. eDirectory scales to the largest network environments, including the Internet. Because it is based on the X.500 standard, eDirectory supports Lightweight Directory Access Protocol (LDAP), Hypertext Transfer Protocol (HTTP), and the Java programming environment.

eDirectory can store and manage millions of objects in a seamless ballet of communications. It also provides the foundation network service for all NetWare servers and network resources. In fact, after network communications, it is the most fundamental network service offered by NetWare 6.

With all this in mind, I'm sure you would agree that eDirectory management is one of your key responsibilities as a Novell CNA (Certified Network Administrator). In this chapter, you will explore four important lessons regarding eDirectory management:

- Introduction to Directory Services—You'll begin with a brief introduction to the basics of Directory services, including some common uses and how a Directory is structured.
- Understanding eDirectory 8.6—Then you'll explore the architecture of Novell eDirectory version 8.6 and compare it to its predecessor, Novell Directory Services (NDS).
- Using eDirectory Objects—You'll dig into the basics behind eDirectory's three types of objects: the Tree object, container objects, and leaf objects.
- Implementing eDirectory 8.6 Naming—Finally, you'll learn about naming conventions used in eDirectory, including naming context rules and inheritance.

As you can see, there's a lot to learn in this chapter and when it's all done, you'll be an accomplished eDirectory administrator. So, let's get started!

Introduction to Directory Services

Test Objectives Covered:

1. Identify basic Directory Service tasks.
2. Identify common Directory Service uses.
3. Describe how a Directory is structured.
4. Identify the role and benefits of eDirectory.

The *Directory Service* is one of the most fundamental network services provided by all NetWare 6 servers. In fact, it represents the communications hub for administrative connectivity between all servers in a large NetWare 6 network. As such, Directory Service management is one of your key responsibilities as the network administrator.

As its name implies, the Directory service provides access to a database, called *eDirectory*, that contains all resources for the entire network. This object-oriented database is organized into a hierarchical structure called the *tree*. eDirectory provides the basic foundation for the Directory service, including capabilities for replication and distribution. *Directory* is capitalized in this case to differentiate it from the directory (or folder) in the file system. In fact, these two "directories" define the two major roles of a NetWare 6 CNA: File System Administrator (directories and files) and eDirectory Administrator (*The* Directory).

The Directory service is your friend. It may seem a little intimidating at first, but when you get to know the Directory service (and eDirectory, for that matter), it's actually fun. Really.

How Directory Services Work

A Directory service classifies all network resources into a finite number of objects. These objects can be organized by function, location, size, type, or color—it doesn't matter. The point is, a Directory service organizes network resources independently from their physical locations. For example, servers are organized according to function. Then users are placed in the appropriate containers to simplify connectivity. This increases productivity because users are near the resources they need. When a user logs in to the network, the user can access any object in the Tree, regardless of its location. This provides a slick means for managing not only users, but also all their hardware and applications. Of course, in the eDirectory tree (as in life), it is best to place User objects and their resources (Printers, Servers, and Volumes) in close proximity to each other.

A Directory service performs several basic tasks, including the following:

- *Connecting disparate systems*—A Directory service integrates and organizes heterogeneous systems to allow them to share common management. In today's business world, such systems are required not only to communicate with each other, but also to share information and use common services to meet the objectives of the organization.
- *Satisfying the needs of the user, organization, and business*—The network must be flexible enough to provide a set of unique services based on individual needs.
- *Emulating all business relationships*—The network must be capable of ensuring that trusted relationships are built and maintained between people, business, the company's intranet, and the World Wide Web.

- *Coordinating information flow*—Information may emanate either from the business (procedural) or from the network (technical). A Directory service coordinates information flow, no matter what the source or type of information.
- *Ensuring information availability*—A Directory service provides a means for making all network information available to users, devices, applications, or other resources.

TIP

A Directory and a Relational Database Management System (RDBMS) are two separate entities with different functions. Even though a Directory is a collection of information, it does not replace the traditional RDBMS. Directories and databases complement one another, even though they serve different purposes.

Typically, a Directory service may be used in the following ways:

- *Organizing data*—A Directory service organizes data or information for the network. In NetWare 6, eDirectory stores all user, server, printer, and other network device information.
- *Accessing information easily*—Similar to the file-and-folder system used on a computer, a Directory service makes information about network resources available to users, devices, and applications. A Directory service provides employees with global access to network resources. Businesses and organizations also use Directory services to provide user authentication and authorization for using these network resources and services. For organizations with large numbers of mobile users, eDirectory provides a means for storing user information required by some applications. (Such applications are described as *Directory-enabled*.) From the user's point of view, a Directory service provides a global view of all network resources, such as users, applications, services, system resources, and devices.
- *Providing security*—A Directory service uniquely identifies network resources, locates network objects when required, supports robust security features, and controls the user access to network resources.
- *Providing services to customers*—For organizations taking advantage of the features of electronic business transactions, Directory services can help organize multiple databases while helping to mesh disparate network systems. This provides better management of processes between customers, employees, and supply-chain partners. The resulting benefits are as follows: reduced costs for administration and hardware, faster access to data and information, and secure network access with superior fault tolerance.

From a general perspective, Directory services can also provide electronic provisioning, enhanced security, customer profiling, electronic wallets, automated notification systems, customized Web interfaces, and virtual private networks (VPNs).

REAL WORLD

A virtual private network (VPN) often is used to transfer sensitive company information across an untrusted network (such as the Internet) in a secure fashion (typically by encapsulating and encrypting data).

So, what do you think? Is a Directory service for you? Who knows—you might even like it.

Directory Architecture

As you recall, the Directory service provides access to the eDirectory database, which contains all resources for the entire network. What exactly does eDirectory look like? From the outside, it looks like a big cloud hovering over your network. On the inside, however, it follows a hierarchical tree structure similar to the Internet domain system. That is, starting at the WWW Root and expanding to ".com" domains and eventually to servers. In NetWare 6, this design is referred to as the *Directory Information Tree*, which is shortened to the "tree" for purposes of our discussion.

Think of the tree as actually being inverted. As in nature, the eDirectory tree starts with the Tree object (called the *Tree Root*) and builds from there. Next, it sprouts container objects, which are branches reaching toward the sky. Finally, leaf objects provide network functionality to users, servers, and the file system. As you can see in Figure 3.1, the tree analogy is alive and well.

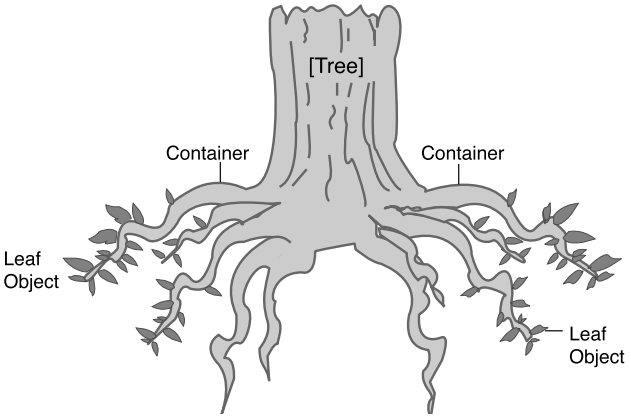

FIGURE 3.1
The figurative Directory services tree.

The real eDirectory tree is made up of logical network objects. eDirectory objects define logical or physical entities that provide organizational or technical function to the network. As you will see later in this chapter, they come in three flavors:

- Tree Root
- Container objects
- Leaf objects

The Tree Root is the very top of the eDirectory tree. Because it represents the opening porthole to the eDirectory world, its icon is appropriately a picture of a tree. Container objects define the organizational boundaries of the eDirectory tree and house other container objects and/or leaf objects. When a container object contains other objects, it is called a *parent object*.

Finally, leaf objects are the physical or logical network resources that provide technical services and network functionality. Leaf objects define the lowest level of the eDirectory structure. You'll learn about the most interesting leaf objects later in this chapter.

The structure of the Directory is governed by a set of rules collectively known as the *Directory schema*. These rules define the type of data, the syntax of that data, and the objects the Directory can contain. Schema rules fall into two broad categories:

- *Object class definitions*—These define the type of objects and the attributes of those objects.
- *Attribute definitions*—These define the structure (syntax and constraints) of an attribute. Simply stated, the attribute value is the actual content or data.

Remember when I told you that eDirectory was based on the X.500 standard? Before you go tree climbing and explore the dynamics of eDirectory, take a quick look at that standard to see what that all means. I think you'll spot some amazing similarities between X.500 and what you've just learned about Directory services.

Understanding X.500

X.500 is an international standard for naming services. A variety of industry standards, such as DNA (Digital Network Architecture), use X.500 with their own naming services to provide address-to-name resolution and directory services. This enables these distributed machines to exist in a large hierarchical management system.

X.500 organizes network resources (such as users and servers) into a globally accessible Directory. The X.500 specification establishes guidelines for representing, accessing, and using information stored in a directory database. In fact, eDirectory is Novell's implementation of the following X.500 features:

- *Scalability*—Large databases can be subdivided into smaller Directory System Agents (DSAs). A DSA can represent either a single organization or multiple organizations, and its contents may be distributed across multiple Directory servers. eDirectory calls them *partitions*.
- *Replication*—This feature allows the Directory database, or portions thereof, to be replicated on backup Directory servers located throughout the network.
- *Synchronization*—Because X.500 must manage a loosely coupled, distributed database, each server must be able to synchronize its database contents with other servers. Directory database updates may be made either at the original master database (master-shadow arrangement) or at any writable replica (peer-to-peer mechanism). In either case, X.500 propagates Directory database change information to all servers holding replicas of the database or a DSA.

The X.500 Directory is represented by a Directory Information Tree (DIT) and Directory Information Base (DIB). At least one of those should sound familiar. The DIB consists of objects (or nodes) and their associated properties and values. Intermediate objects act as containers that aid in organizing the DIT. Leaf objects represent individual network entities, such as servers, printers, and so on. Refer to Figure 3.2 for an illustration of the X.500 Directory architecture.

The rules that determine the type of information that may be stored in the DIB are held in the Directory's schema. (Now this should be sounding really familiar.) Each object in an X.500 DIT has a unique name that is referred to as its distinguished name, or DN, (that is, complete name). Each object may also be referred to by a relative distinguished name, or RDN, (that is, partial name).

Directory database access is managed by a DSA running on a local server. Users access the database through a Directory User Agent (DUA). DUAs are available in command-line, forms-based, and browser-style interfaces. DSAs and DUAs communicate with each other using the Directory Access Protocol (DAP). Furthermore, DSAs may communicate with one another using the Directory System Protocol (DSP), Directory Information Shadowing Protocol (DISP), or the Directory Operational Binding Management Protocol (DOP).

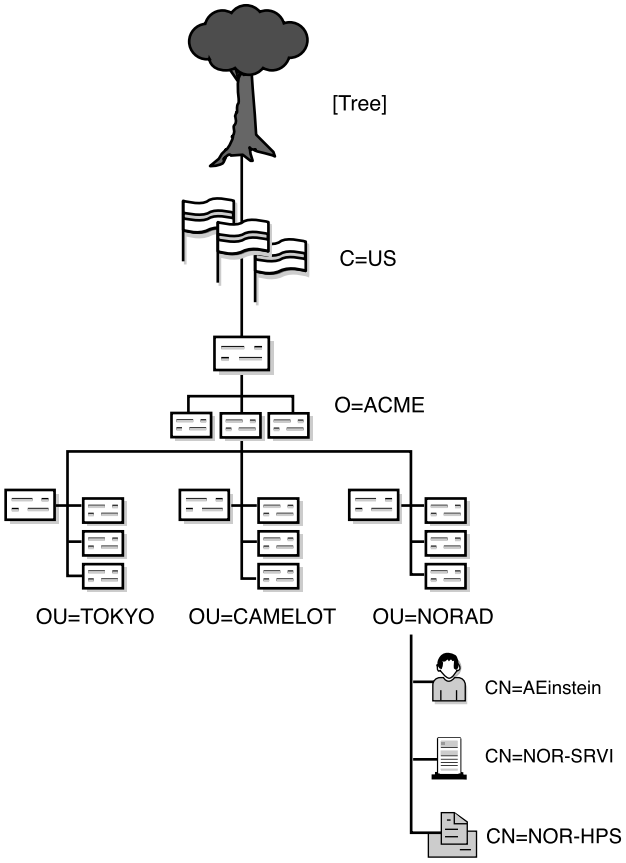

FIGURE 3.2
X.500 Directory architecture.

Now that you know where Directory services came from and generally how they work in NetWare 6, it's time to do some tree climbing! Sounds like that fun I promised you, doesn't it?

Understanding eDirectory 8.6

Test Objectives Covered:

4. Identify the role and benefits of eDirectory (*continued*).
5. Identify how eDirectory 8.6 works.
6. Identify and describe the composition of eDirectory.

eDirectory 8.6 is a highly scalable, high-performing, secure Directory service. Along with replication and partitioning capabilities, eDirectory provides

the basic foundation for multiplatform networking. eDirectory also includes cryptography services to protect confidential data; it natively supports LDAP 3 over Secure Socket Layer (SSL).

Although all NetWare 6 servers on the network use the Directory, you probably don't want to store a complete copy of it on each server. This is particularly true if you have a large network. Fortunately, NetWare 6 enables you to break up the Directory into smaller pieces called *partitions* and replicate them on multiple servers. This means that any NetWare 6 server can contain a copy of the entire Directory, specific pieces of it (partitions), or none at all. Of course, it's best to keep copies of important partitions closest to the users who need them. This minimizes unnecessary replica synchronization and background network traffic.

Features and Benefits of eDirectory 8.6

Following are some reasons why I think eDirectory is the greatest thing since sliced bread:

- eDirectory offers a global database for central access and management of network information, resources, and services.
- eDirectory offers a standard method of managing, viewing, and accessing network information, resources, and services.
- eDirectory enables you to logically organize your resources independent from their physical characteristics or layout of the network.
- eDirectory provides dynamic mapping between an object and the physical resource to which it refers.
- eDirectory works today and is several years ahead of any competitor with proven reliability, scalability, and security for enterprise networks.
- eDirectory significantly lowers the cost of managing and administering a network through centralized access and management of all network and operating system resources. In addition, it significantly lowers the cost of connectivity and data synchronization over a wide area network.

The eDirectory architecture, which you'll examine in detail later in this section, provides an exceptional foundation for all of eDirectory 8.6's new features and benefits.

Following is a brief list of some of eDirectory's greatest new advancements:

- eDirectory 8.6 can be implemented on any of these operating system platforms: NetWare, Windows NT, Windows 2000, Linux, Solaris, and Tru64 UNIX. Client libraries and LDAP tools are available for Linux, Solaris, and Tru64 UNIX. LDAP support provides an open structure for integration with applications that are written to the LDAP standard.
- The Index Manager tool enables you to manage database indexes easily. The Filtered Replica Configuration Wizard enables you to easily create filtered replicas, which are replicas that contain a filtered list of network resources.
- The eDirectory Import/Export Wizard enables you to import or export LDIF files and to perform a server-to-server data migration.
- eDirectory includes a merge utility that enables you to merge one directory tree into another or to graft one tree onto another.
- iMonitor provides monitoring and diagnostic capabilities for all servers in your eDirectory tree from a Web browser.
- ConsoleOne provides you with a utility to manage eDirectory users, objects, schema, partitions, and replicas.

Some of the major benefits of eDirectory are as follows:

- Central management of network information, resources, and services
- Standard method of managing, viewing, and accessing network information, resources, and services
- Logical organization of network resources that are independent of the physical characteristics or layout of the network
- Dynamic mapping between an object and the physical resource to which it refers

The Role of eDirectory

When a NetWare client (such as a user, application, or server) requests access to a network resource or service, eDirectory satisfies the request according to data stored in the network-wide Directory. One of the advantages of this strategy is that client requests are separated from resource physicality—that is, users don't need to know where a physical resource is located. They simply reference its unique Directory name. Because all NetWare 6 servers provide eDirectory, any NetWare 6 server on the same network can connect you with the resource.

The following list describes how eDirectory processes client requests:

1. The user logs in via a NetWare 6 client and establishes credentials and signature. A *credential* is a data structure such as a network address, time of login, username, and password. It consists of a validation period and other identification information. A *signature* is the result of encryption of this data.
2. The NetWare 6 client requests a service that has been requested by a user or application. The service responds by sending the client a random number generated for the current transaction only. (The random number is not used again.)
3. Using the signature, the client provides proof that the credential and random number are correct.
4. The client sends the random number, the credential, and the signature to the service.
5. Client validity and authority are checked by verifying that the proof was legitimately generated from the random number and credential. The random number ensures that the request was created from the current session.
6. The service returns a confirmation.
7. The client is then connected to the resource.

Earlier versions of eDirectory were called Novell Directory Services (NDS). At first glance, eDirectory appears to have the same underlying architecture as NDS—that is, a distributed, object-oriented database organized as a hierarchical tree. Upon closer inspection, however, you'll find that eDirectory 8.6 is built on a much more sophisticated database structure than NDS.

Let's take a closer look at the underlying architectural differences between NDS and eDirectory, starting with NDS.

NDS Architecture

NDS was first introduced in NetWare 4. Prior to NetWare 4, NetWare operating systems relied on a server-centric model in which each NetWare server had its own flat-file database for tracking network resources (called the Bindery). The bindery consisted of three files: one that held objects, one that held property, and one that held value information.

NDS offered a gigantic leap forward by evolving the server-centric model into a network-centric model. In this architecture (shown in Figure 3.3), the

NetWare 4 operating system relies on four data files and multiple streams files located in a hidden directory on the server's SYS: volume. This database is called the RECMAN database.

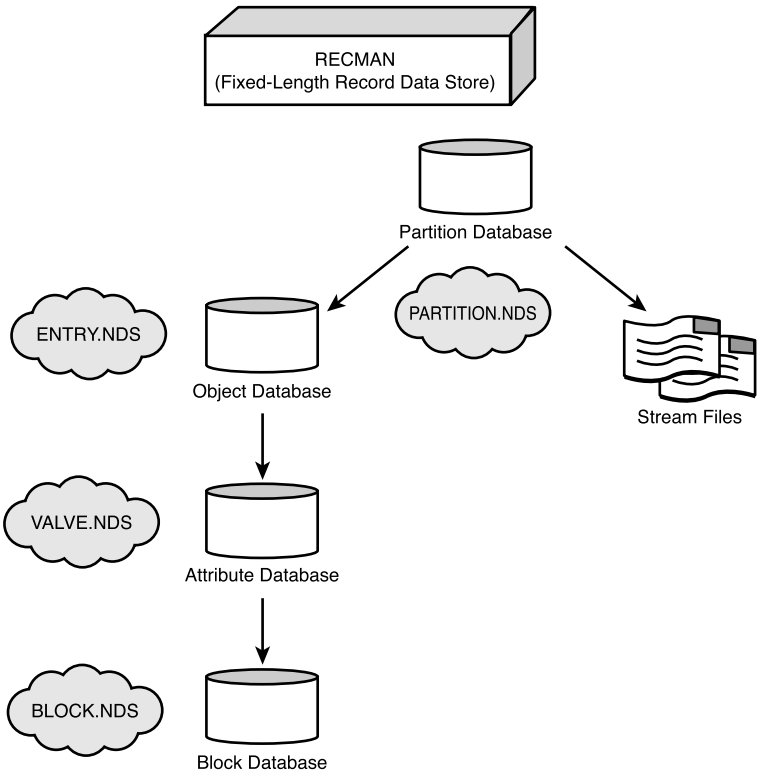

FIGURE 3.3
NDS architecture.

The four files that make up the NDS architecture in Figure 3.3 perform the following functions:

- PARTITIO.NDS—The partition database contains a list of database partitions including system, schema, external reference, and bindery.
- ENTRY.NDS—The object database contains records for each object in a given server's replicas.
- VALUE.NDS—The attribute database contains property values for each object in ENTRY.NDS.
- BLOCK.NDS—The block database contains overflow data for the attribute database.

NDS streams files are named with hexadecimal characters (0–9, A–F) and hold information such as print job configurations and login scripts. Earlier

versions of NDS used Novell's Transactional Tracking System (TTS) to ensure that database transactions were either completed or backed out in the event of a system failure.

TIP

The NetWare 5 version of NDS uses the same architecture as described previously; however, the names of the files are different. In NetWare 5, ENTRY.NDS is called 0.DSD, VALUE.NDS is called 1.DSD, BLOCK.NDS is called 2.DSD, and PARTITIO.NDS is called 3.DSD.

eDirectory 8.6 Architecture

eDirectory 8.6 improves on NDS's fixed-length record data store model by introducing a highly scalable indexed database called the FLexible and Adaptable Information Manager (FLAIM). The FLAIM database uses three types of files instead of four, but still relies on streams files for print job configurations and login scripts. Check out the eDirectory 8.6 architecture shown in Figure 3.4.

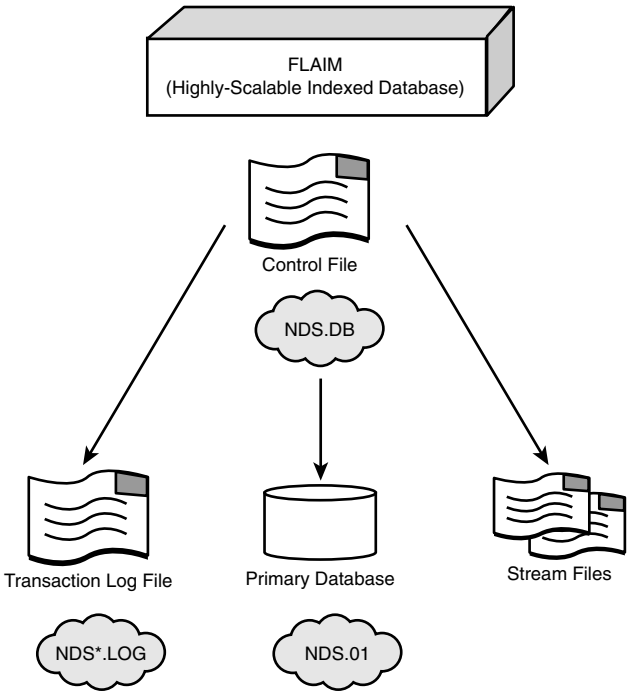

FIGURE 3.4
eDirectory 8.6 architecture.

Following is a description of each of the three types of files that make up eDirectory's FLAIM database:

- NDS.DB—The control file is the centerpiece of the eDirectory architecture. This file contains the rollback log and is used to abort incomplete transactions.
- NDS.01—The primary database file contains all records and indexes found on a given server. When this data file reaches 2GB in size, NDS.02 is created for the remaining data. New files are then created as necessary to keep database files from growing beyond 2GB. This allows the database files to remain scalable while retaining their quick search capabilities.
- NDS*.LOG—The transaction log file acts as a roll-forward log to reapply completed transactions that might not have been fully written to disk because of a system interruption.

eDirectory streams files perform the same function that they do in NDS and have an .NDS extension. However, unlike NDS, eDirectory does not use TTS; instead, it uses log files to back out and roll forward transactions in the event of a system failure. Refer to Table 3.1 for a summary of the differences between NDS and eDirectory architecture.

TIP

The primary eDirectory database file, NDS.01, includes a number of indexes to enhance performance. First, it includes attribute substring indexes for the CN and uniqueID fields. Second, it includes attribute indexes for the Object Class and dc fields. Finally, it includes attribute indexes for positioning that include strings beginning with CN, uniqueID, Given Name, and Surname.

TABLE 3.1

Comparing NDS and eDirectory Architecture

COMPONENT	NDS	EDIRECTORY
Database Name	RECMAN	FLAIM
Database Function	Fixed-Length Record Data Store	Highly Scaleable Indexed Database
NetWare Version	4.x and 5.x	6.0
Number of Files	4	3
Data Records File	ENTRY.NDS	NDS.01
Rollback Mechanism	TTS	Log Files
Streams	Yes	Yes

This completes the architectural lesson of eDirectory 8.6. We hope that you have gained an appreciation for the sophisticated directory services platform that eDirectory 8.6 provides for your NetWare 6 network. Now that you understand how it's built, you're ready to learn how to integrate it into your existing network.

Now that you understand the fundamental architecture of eDirectory, the next sections take a closer look at its different container and leaf objects. These are the physical foundation of the logical eDirectory tree.

Using eDirectory Objects

Test Objectives Covered:

7. Identify and describe eDirectory object classes.
8. Identify the flow and design of the eDirectory tree.

The Directory consists of the schema, objects, properties, and values.

The *schema* defines the types of objects that you can create, as well as what information is required when the object is created. Each type of object has associated with it a defined schema class. NetWare 6 ships with the *base schema*, which when modified becomes known as an *extended schema*. Modifications to the schema usually are done with the Schema Manager in the ConsoleOne utility.

An *object* is similar to a record or row of information in a database table. It contains information about a particular network entity. eDirectory represents each network resource as an object in the Directory. For example, a User object represents a particular user on the network. An object can be a physical resource (such as a workstation), an eDirectory resource (such as a group), or an organizational resource (such as a container).

An object *property* is similar to a field in a database record. It is a category of information you can store about an object. For example, properties of a User object include such things as Login Name, Password, and Telephone Number. Each type of object has a specific set of properties associated with it; this defines its *class*. Properties are predefined by eDirectory and determine how a given object can be used. For example, Server properties differ from Printer properties because they are different eDirectory objects with different functions.

Three important types of eDirectory properties are the following:

- *Required properties*—These properties contain vital object data and, therefore, must be supplied to create the object. Required properties can't be deleted. For example, when you create a User object, you must indicate values for the Login Name and Last Name properties. In fact, the name of an object is always a required property. Otherwise, you would have no way of referring to it.
- *Optional properties*—These properties contain nonvital information about an object. As such, you need to supply values for them only if desired. Examples include a User's Title, Telephone Number, and Fax Number.
- *Multivalued properties*—These properties support more than one entry. For example, the Telephone Number property associated with a User object can hold multiple phone numbers for that user. Other User-related multivalued properties include Title, Location, and Fax Number. (This type of multivalued property is represented in ConsoleOne with an ellipsis (…) button to the right of the property field. If you click this button, you'll be allowed to enter additional entries for the property.)

Finally, a property *value* is similar to a data string in a field of a database record. In other words, it's a data item associated with a property. For example, the value associated with the Password property of a User object would be the actual password for that User object.

Refer to Table 3.2 for an illustration of the relationship between eDirectory objects, properties, and values.

TABLE 3.2 **Understanding eDirectory Objects, Properties, and Values**

OBJECT	PROPERTY	VALUE
User	Login Name	AEinstein (also known as AEinstein.LABS.NORAD.ACME)
	Title	Super Smart Scientist
	Location	NORAD
	Password	relativity
Printer (Non NDPS)	Name	WHITE-P1.WHITE.CRIME.TOKYO.ACME

Table 3.2 Continued

OBJECT	PROPERTY	VALUE
	Default Print Queue	WHITE-PQ1.WHITE.CRIME.TOKYO.ACME
	Print Server	WHITE-PS1.WHITE.CRIME.TOKYO.ACME
NetWare Server	Other Name	LABS-SRV1
	Version	Novell NetWare 6.01g[DS]
	Operators	Admin
	Status	Up

Hierarchy of eDirectory

As you learned earlier, the Directory is an object-oriented database that is organized in a hierarchical structure called the eDirectory tree. It provides a way to view the logical organization of network resources stored in the Directory database. As you can see in Figure 3.5, the tree is similar to the DOS file system.

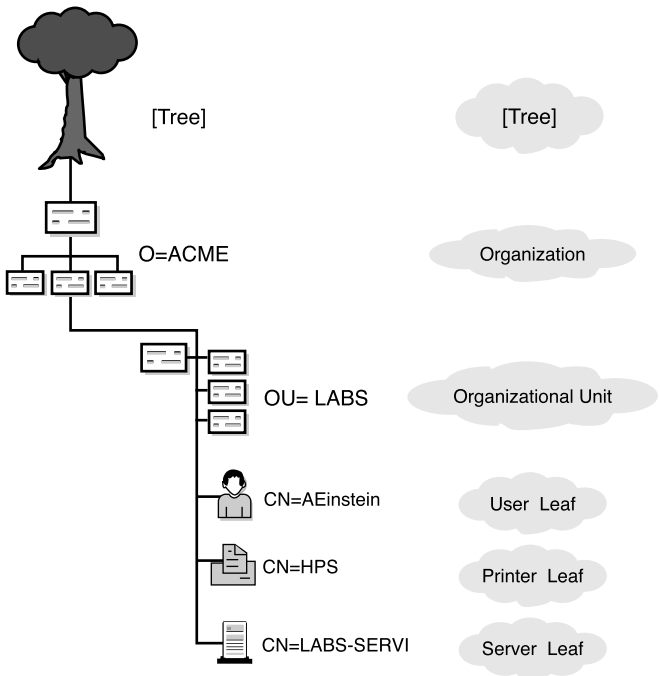

FIGURE 3.5 Understanding eDirectory objects.

The schema contains object classes, which represent the actual definition of each type of eDirectory object. eDirectory contains the following three main classes of objects:

- Tree Root
- Container objects
- Leaf objects

The top of the tree is called the *Tree Root*. *Container objects*, which are analogous to folders, define the organizational boundaries of the Directory and are used to store other container objects and/or leaf objects, depending on which type of container they are. (A container object is called a *parent object* if it contains other objects.) *Leaf objects*, which are analogous to files, are typically stored in container objects. They are the physical or logical network resources that provide technical services and WAN functionality. Leaf objects define the lowest level of the eDirectory structure and therefore cannot contain other objects.

The main difference between eDirectory and DOS architecture is that eDirectory containers have restrictions on where they can be placed and what can be placed in them. Typically, each NetWare 6 network will have only one eDirectory tree. If a network has more than one tree, each will function as a separate, independent database. In other words, resources cannot be shared between them.

In the Directory, each network resource is defined as a logical object. There are a number of types of objects. For example, an object can represent a person (such as a user), a physical resource (such as a printer), an eDirectory resource (such as a group), or an organizational resource (such as an Organizational Unit container).

The bottom line is this—users don't access physical resources anymore. Instead, they access logical objects in the eDirectory tree. This means they don't need to know which NetWare 6 server provides a particular resource. All they need to know is where the resource exists in the logical eDirectory world.

Now that you've mastered the subtle differences between eDirectory schema, objects, properties, and values, let's explore some of the most interesting objects in detail, starting at the top with the Tree Root.

TIP

One of the best ways to remember the nuances of eDirectory hierarchy is to follow the yellow brick road to the land of 3s:

- **3 Main eDirectory Components in the Schema: Object, Property, Value**
- **3 Main Classes of Objects: Tree, Container, Leaf**
- **3 Main Container Objects: Country (C), Organization (O), Organizational Unit (OU)**

Tree Root

The Tree object (also known as the *Tree Root*) is a required object that defines the top of the eDirectory organizational structure. Because it represents the opening porthole to the eDirectory world, its icon is appropriately a picture of a tree. Each Directory tree can have only one Tree Root, which is created during installation of the first server in that tree. The only objects that can be created directly under the Tree are Country, Organization, and Alias. (In this case, the Alias object can point only to a Country or Organization.)

Although some people think of the Tree as a container object (because it contains all the objects in the Directory), it differs from other container objects in the following ways:

- It cannot be created except during installation of the first NetWare 6 server on a network.
- It is essentially a placeholder; it has only one property: Name. The Tree name is shown in the hierarchy of ConsoleOne.
- It cannot be moved, deleted, or renamed.
- It uses the eDirectory tree name, which can be changed at a later date.

Like other objects, the Tree can be assigned as a trustee of other objects, and other objects can be granted trustee access rights to it. For example, an object can be granted trustee rights to the entire eDirectory tree by making the object a trustee of the Tree object. (See Chapter 6, "NetWare 6 Security," for further information on trustee rights.)

Container Objects

Container objects are logical objects that organize (store) other container or leaf objects. A container can represent a country, a location within a country, a company, a department, a responsibility center, a workgroup, or a

collection of shared resources. Each class of container object has different rules that define what it can contain and where it can be located in the tree. Each class also has different properties.

The following are the three most common types of NetWare 6 container objects:

- *Country*—Designates the country where certain parts of the organization reside.
- *Organization*—Represents a company, university, or department. eDirectory supports only one layer of Organization objects; hence, the term *one-dimensional*. Organizations can hold Organizational Unit containers or Leaf objects.
- *Organizational Unit*—Represents a division, a business unit, or a project team within the Organization. Organizational Units hold other Organizational Units or leaf objects. They are *multidimensional*.

Refer to Figure 3.5 earlier in this chapter for an illustration of the relationship between the Tree Root and container objects. The ACME Organization houses other Organizational Units (including LABS), that in turn house leaf objects (like AEinstein). In the next sections, you'll take a closer look at these three container objects.

Country

The *Country* object is an optional container that organizes a Directory tree by country. This type of object can be defined only directly under the Tree Root and must be named using a two-character abbreviation. Novell states that you must use a valid two-character country abbreviation. Presumably, this is to ensure that your network is in compliance with the two-character abbreviations defined in the ISO X.500 standard.

Interestingly, if you create a Country object using the ConsoleOne utility, it allows you to use any two-character name. To determine which two-character names are compliant with the ISO X.500 standard, click Help when creating the Country object, and NetWare 6 will tell you. You can also visit the ISO Web site at `www.iso.ch/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1.html` to see a list of country codes. The only objects that can exist in a Country container are an Organization or Alias object pointing to an Organization.

TIP

If you don't have any compelling reasons to use the Country object, stay away from it. It adds an unnecessary level of complexity to your network. In fact, Novell doesn't even use the Country object in its own multidimensional, worldwide eDirectory tree.

Organization

If you don't use a Country object, the next layer in the tree is typically an Organization. You can use an Organization object to designate a company, a division of a company, a university or college with various departments, and so on. Every Directory tree must contain at least one Organization object. Therefore, at least one Organization is required. Many small implementations use only the Organization object and place all their resources directly underneath it. Organization objects must be placed directly below the Tree Root, unless a Country object is used. Finally, Organization objects can contain all objects except Tree Root, Country, and Organization.

Earlier, we defined the Organization as a one-dimensional object. This means the tree can support only one layer of Organization objects. If you look closer at the icon, you'll see a box with multiple horizontal boxes underneath. Additional vertical hierarchy is defined by Organizational Units, which are multidimensional. You'll learn about them in just a moment.

Figure 3.6 illustrates the object dialog box for the ACME Organization using ConsoleOne. On the right side of the screen are the many page buttons that identify categories of eDirectory properties for this object. Associated with each page button is an input screen for a specific set of information (on the left side). The Identification page button (shown here) allows you to define a variety of Organization properties, including Name, Other Name, Description, Location, Telephone, and Fax Number.

Similar page buttons enable you to configure important Organization parameters, including postal information, print job configurations, trustee assignments, and so on. As far as ACME is concerned, the Organization container defines the top of the functional tree.

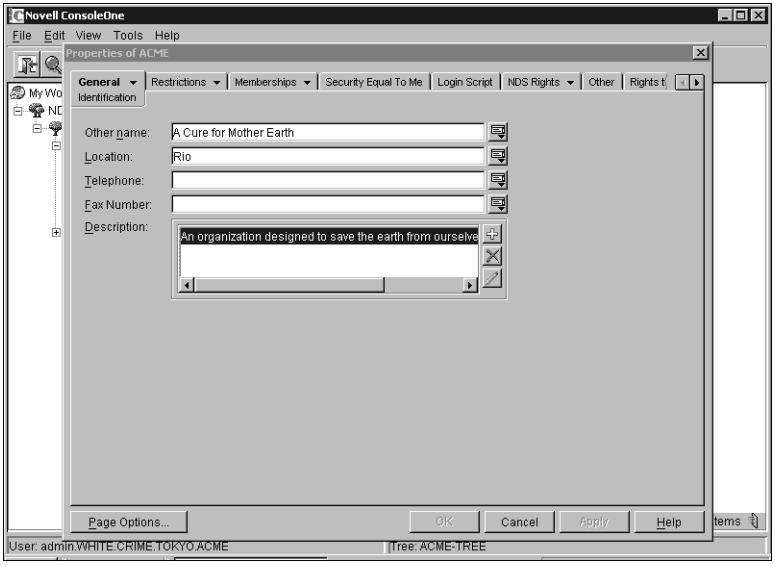

FIGURE 3.6
Properties of an eDirectory Organization object.

Organizational Unit

The Organizational Unit object is a natural group. It enables you to organize users with the leaf objects they use. You can create group login scripts, a user template for security, trustee assignments, security equivalences, and distributed administrators. Although the Organizational Unit container is optional, you should definitely use as many as you need to build a scalable hierarchy for two main goals: resource access and partitioning.

Organizational Units can represent a division, a business unit, a project team, or a department. Organizational Units are multidimensional, in that you can have many hierarchical levels of containers within containers. Remember, Organization objects can exist only at one level in the eDirectory tree.

Organizational Units are the most flexible containers because they can contain other Organizational Unit objects or leaf objects. Organizational Units can contain most of the eDirectory object types, except the Tree Root, Country, or Organization containers (or Aliases of any of these).

Now take a look at the real stars of the eDirectory world—leaf objects.

REAL WORLD

eDirectory supports many other special-use container objects. Five that you may frequently run across are Locality (L), Domain (DC), License Container (LC), Security Container (S), and Role-Based Service (RBS).

The "secret" Locality object is similar to the Country object, in that it's optional and not created as part of the default NetWare 6 installation. If desired, you can use a Locality object to designate the region where your organization's headquarters resides. Unlike Country objects, Locality objects can reside either under the Country, Organization, or Organizational Unit containers.

In addition, NetWare 6 supports three types of administration container objects: a License Container, a Security Container, and a Role-Based Service container. These objects are added to the eDirectory tree when NetWare 6 and Novell Licensing Services (NLS) are installed. They appear as eDirectory objects in the container that includes the Server object. License Container objects can contain multiple-license Certificate leaf objects. Security Containers hold global login and authentication policies, whereas RBS containers hold Role leaf objects for distributing key administration tasks to groups of users.

Finally, Domain containers enable you to use your Domain Name System (DNS) to help locate services in the eDirectory tree. Domain objects can reside directly under the Tree Root, or within Country, Organization, Organizational Unit, or Locality containers.

Leaf Objects

Leaf objects represent logical or physical network resources. Because leaf objects reside at the end of the structural eDirectory tree, they cannot be used to store other objects. In other words, they represent the proverbial "end of the road." As you learned earlier, each class of leaf object has certain properties associated with it. This collection of properties differentiates the various leaf object classes from each other. For example, User objects contain different properties than Printer objects do.

The following are some of the key leaf objects covered in this course:

- *Alias*—An Alias object points to another object that exists in a different location in the eDirectory tree. It enables a user to access an object outside of the user's normal working context (that is, container). An Alias object does not carry trustee rights.

- *Application*—An Application object enables network administrators to manage applications as objects in the eDirectory tree. The advantage of this object is that users don't have to worry about drive mappings, paths, or rights when they want to execute an application. This information is defined by Application object properties.

- *Directory Map*—A Directory Map object represents a logical pointer to a physical directory or folder in the NetWare 6 file system. This object is useful in mapping statements because it enables you to map a drive to a resource without knowing its physical location. If the path to the resource changes, you need to change only the path designated in the Directory Map object, rather than any of the login script mappings statements that refer to it.

- *Group*—A Group object defines a list of users for the purpose of assigning access rights or other configuration parameters. The members of a group can be a subset of users in the same container or spread across multiple containers. The difference between containers and groups is that container objects store User objects, whereas Group objects store a list of User objects. Groups enable one of the great CNA tricks: they allow you to specify login script commands to a subset of users with the IF MEMBER OF syntax.

- *LDAP Group*—An LDAP Group object stores configuration data (class mappings, attribute mappings, and security policies) for groups of LDAP Servers. During installation, an LDAP Group object named LDAP Group- <*servername*> is created by default in the host Server's home container.

- *LDAP Server*—An LDAP Server object stores configuration data for a NetWare 6 server running LDAP Servers for eDirectory. During installation, an LDAP Server object named LDAP Server- <*servername*> is created by default in the host Server's home container. Make sure not to assign the same LDAP Server object to more than one server running LDAP Services for eDirectory.

- *License Certificate*—A License Certificate object is used by NetWare Licensing Services (NLS) to monitor and control the use of licensed applications on the network. When the NLS-aware application is installed, a License Certificate object is added to the Licensed Product container.

- *NDPS Broker*—An NDPS Broker provides three network support services for network printing: Service Registry Services (SRS), Event Notification Services (ENS), and Resource Management Services (RMS). When NDPS is installed, the installation utility ensures that a Broker object is loaded on your network.

- *NDPS Manager*—The NDPS Manager is a logical entity used to create and manage NDPS Printers and Printer Agents. The NDPS Manager object stores information used by NDPSM.NLM on a single server. This single server can control multiple Printer Agents.

- *NDPS Printer*—NDPS Printers magically transform electronic bits into written words. These objects are created by iManager as *Controlled Access Printers*. As eDirectory objects, NDPS Printers are no longer available as Public Access Printers.

- *Organizational Role*—An Organizational Role object defines a position or role within the organization that can be filled by any designated user. The Organizational Role is particularly useful for rotating positions that support multiple employees, where the responsibilities of the job, and the network access required, are static. If a User object is assigned as an *occupant* of an organizational role, the user "absorbs" all trustee rights assigned to it. Some organizational role examples include PostMaster, Network Administrator, Silicon Valley CEO, or Receptionist.

- *Print Server (Non NDPS)*—A Print Server (Non NDPS) object represents a network print server used for monitoring queue-based print queues and printers.

- *Printer (Non NDPS)*—A Printer (Non NDPS) object represents a queue-based physical printing device on the network, such as a printer or plotter. Refer to Figure 3.7 for an illustration of the Printer (Non NDPS) properties that can be managed using ConsoleOne.

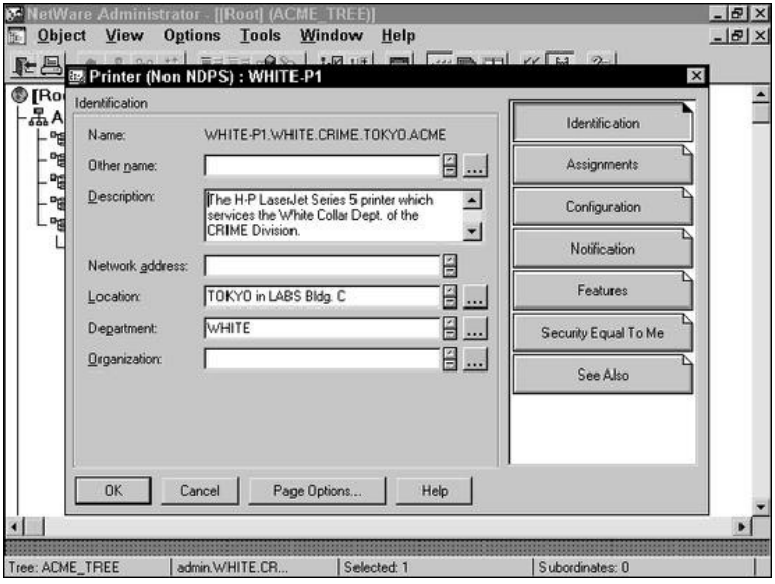

FIGURE 3.7
Properties of an eDirectory Printer (Non NDPS) object.

- *Profile*—A Profile object defines a login script for a subset of users in the same container or spread across multiple containers. (If all the users in a container need the same login script, you should use a Container login script, instead.)

- *Server*—A Server object represents a server running eDirectory on your network. eDirectory supports the following operating system platforms: NetWare, Solaris, Linux, and/or Windows 2000. You can even create a Server object for Bindery servers running NetWare 2 or NetWare 3. This object is used by various leaf objects (such as Volume objects) to identify a physical server that provides particular network services. Refer to Figure 3.8 for an illustration of the Server properties that can be managed using ConsoleOne.

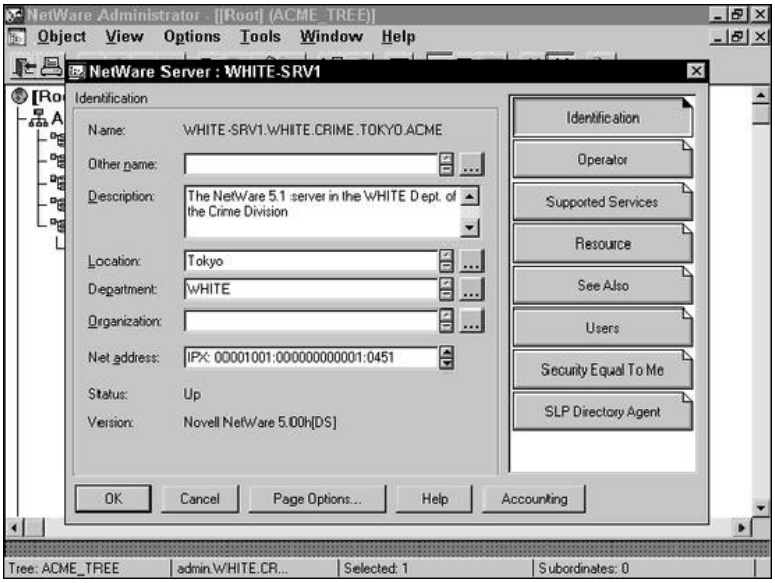

FIGURE 3.8
Properties of an eDirectory Server object.

- *Unknown*—An Unknown object represents an eDirectory object that has been corrupted, invalidated, or that cannot be identified as belonging to any of the other leaf classes. For example, an Alias object becomes Unknown when its host is deleted. Ouch!

- *User*—A User object represents a person who uses the network (for example, you, me, or Fred). For security reasons, you should create a separate User object for each user on the network. A User object contains a plethora of interesting properties, including Login Name, Password, Full Name, Login Restrictions, and so on. Refer to Figure 3.9 for an illustration of the User properties that can be managed using ConsoleOne.

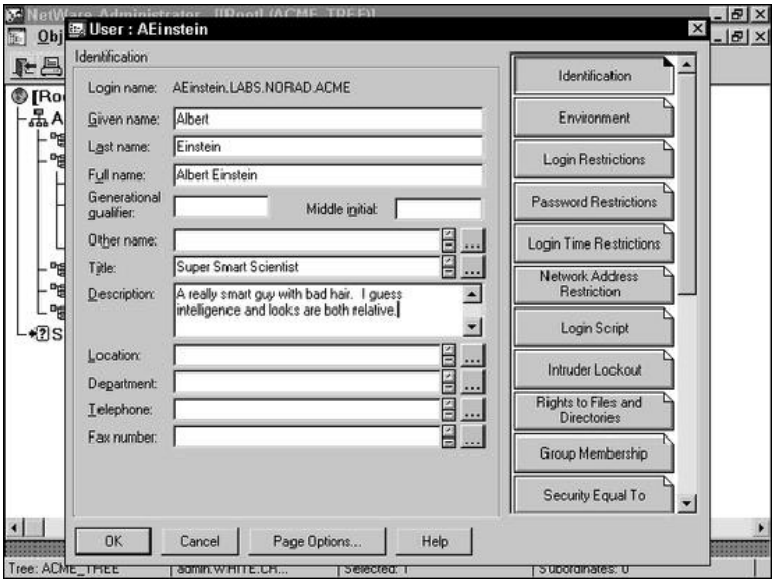

FIGURE 3.9
Properties of an eDirectory User object.

REAL WORLD

eDirectory is a powerful database with much valuable user information. Consider using it as a central company database of employee data. If you can't find what you need in the almost 100 default properties, you can always create your own. The NetWare 6 Software Developers Kit (SDK) provides interface tools for modifying and adding eDirectory properties. This is called *extending* the eDirectory Schema.

In addition, NetWare 6 includes a Schema Manager for viewing and customizing the eDirectory Schema directly from ConsoleOne. You can access this GUI (graphical user interface) management tool from the Tools menu. Check it out…it's fun at parties!

- *Volume*—A Volume object represents a physical volume on the network. Typically, volumes are created during the server installation process. Remember that a Volume object is a leaf object rather than a container object—even though the ConsoleOne utility may give you the opposite impression. It stores information about a volume, including server name, physical volume mapping, and volume use statistics. No information about files and folders on a volume is contained in a Volume object. However, you can access that information through ConsoleOne. Volume objects are supported only on NetWare. If you are using UNIX file system partitions, these cannot be managed using Volume objects.

- *Workstation*—A Workstation object enables you to manage network workstations through eDirectory. This leaf object is automatically created when a workstation is registered and imported into the eDirectory tree.

Table 3.3 summarizes some important eDirectory object characteristics.

TABLE 3.3 eDirectory Object Characteristics

OBJECT	CAN EXIST IN	CAN CONTAIN	EXAMPLES
Tree Root *(Required)*	Top of the tree	Country Organization Alias *(of Country or Organization only)*	ACME_TREE
Country *(Optional)*	Tree Root	Organization Alias *(of Organization only)*	US UK
Organization *(Required)*	Tree Root Country	Organizational Units All leaf objects	Novell MIT
Organizational Unit *(Optional)*	Organization Organizational Unit	Organizational Units All leaf objects	Sales Finance
Leaf objects *(Required and Optional)*	Tree Root (*Alias of Country or Organization only*) Country (*Alias of Organization only*) Organization Organizational Unit	Cannot contain other eDirectory objects	CRIME-SRV1 DClarkeIV KShafer

This completes the discussion of the most interesting eDirectory objects offered by NetWare 6. You'll want to get to know all these leaf objects because future discussions center around how to organize, design, and manage them. After you understand the relationships between eDirectory objects, you can start building your tree.

TIP

Every leaf object and container object is represented by an icon graphic that depicts its purpose. For example, printers are printers, servers are computers, and users are people. These icons are used throughout this book and in graphical eDirectory utilities. ConsoleOne, for example, uses icons to provide a snapshot of an entire eDirectory tree in a hierarchical structure. This feature makes it easier for administrators and users to locate and use NetWare 6 resources.

How do you feel? So far, you've explored all the physical and logical objects that make up NetWare 6's revolutionary eDirectory tree. However, this is only the beginning. Your success as a CNA is defined by your ability to manage eDirectory objects and their properties.

Lab Exercise 3.1: Getting to Know eDirectory

Part I

Write C for container or L for leaf next to each of the following objects:

1. ___Volume
2. ___Country
3. ___User
4. ___Group
5. ___Organizational Unit
6. ___Server
7. ___Print Queue
8. ___Organizational Role
9. ___NDPS Broker
10. ___Organization

See Appendix C for answers.

PART II

Indicate whether you think each item below would be a container or a leaf object. If you think it would be a container object, indicate what type of container (that is, Country, Organization, or Organizational Unit).

1. _____The Human Resources Department
2. _____David IV
3. _____A database server
4. _____The PAYCHECK print queue
5. _____ACME, Inc.
6. _____The Administrator Organizational Role

7. _____UK (that is, United Kingdom)
8. _____A dot matrix printer
9. _____The Tokyo office
10. _____The SYS: volume

See Appendix C for answers.

Implementing eDirectory Naming

Test Objective Covered:

8. Identify the flow and design of the eDirectory tree *(continued)*.

Now that you understand what the eDirectory tree is made of, you need to explore how it works. As you manage the eDirectory tree, pay particular attention to its structure. A well-designed tree will make resource access and management much easier.

The structure of the eDirectory tree is both organizational and functional. The location of an object in the tree can affect how users access it and how network administrators manage it. The eDirectory tree structure impacts the following areas of administrative responsibility:

- eDirectory planning
- Resource access
- Resource setup

You complete these tasks by implementing eDirectory planning guidelines, using proper eDirectory naming structure, and understanding current context. Let's take a closer look.

Planning Guidelines

An efficient eDirectory tree provides all the following benefits:

- It makes resource access easier for users.
- It makes administration easier for network administrators.
- It provides fault tolerance for the eDirectory database.
- It decreases network traffic.

The structure of the tree can be based on location, organization, or administration. In many cases, it's a combination of all three. Many factors influence the structure of your eDirectory tree. Before you design your tree, you might need to study workgroups, resource allocation, and/or learn how data flows throughout your network.

As a CNA, it's your responsibility to navigate and manage the tree, not to design or troubleshoot it—that's what CNEs are for. This material is also covered in much greater detail in *Novell's CNE Study Guide for NetWare 6*.

TIP

Study the eDirectory benefits carefully—they are the foundation of your life as a NetWare 6 CNA. One final important point: eDirectory does *not* provide fault tolerance for the file system.

eDirectory Naming Structure

eDirectory naming defines rules for locating leaf objects. One of the most important aspects of a leaf object is its position in the eDirectory tree. Proper naming is required when logging in, accessing eDirectory utilities, printing, and for most other management tasks.

The name of an eDirectory object identifies its location in the hierarchical tree. Therefore, each object name must be unique. eDirectory naming impacts two important NetWare 6 tasks:

- *Login*—Typically, you need to identify the location of your User object in the eDirectory tree for NetWare 6 to authenticate you during login.
- *Resource access*—eDirectory naming exactly identifies the type and location of NetWare 6 resources, including file servers, printers, login scripts, and files.

The whole NetWare 6 eDirectory naming scheme is much more complicated than "Hi, I'm Fred." It requires both your name and location. For example, a proper eDirectory name would be "Hi, I'm Fred in the ADMIN division of ACME." As you can see in Figure 3.10, Fred's eDirectory name identifies who he is and where he works.

The eDirectory tree affects resource access because the organization of objects in the tree dictates how they can be found and used. In fact, the whole eDirectory naming strategy hinges on the concept of *context*. Context defines the position of an object within the Directory tree structure. When you request a particular network resource, you must identify the object's context so that eDirectory can find it. Similar to locating files by using a DOS directory path, context represents a list of container objects leading from the object to the Tree Root. NetWare 6 uses specific naming guidelines for creating an object's context.

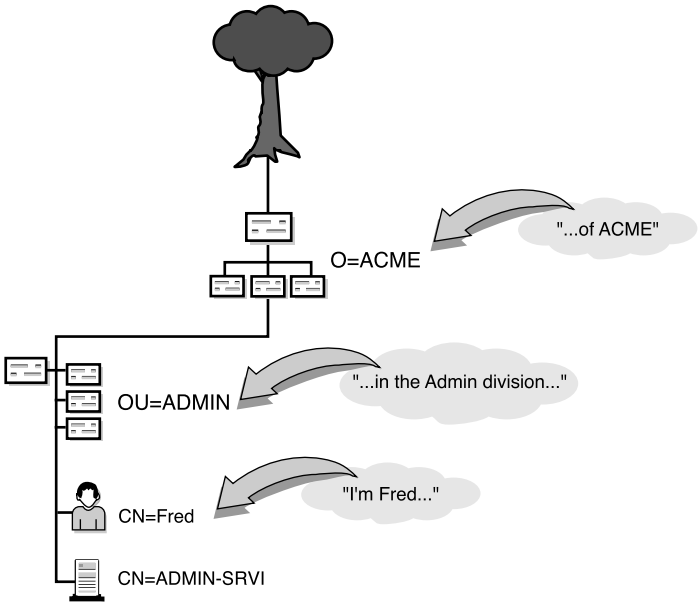

FIGURE 3.10
Getting to know the real Fred.

REAL WORLD

Novell recommends that before you implement eDirectory, you create a document that describes your naming standards. The eDirectory naming rules you're going to learn here work only if object names are consistent across the network. A naming standards document provides guidelines for naming key container and leaf objects, including users, printers, servers, volumes, print queues, and organizational units. In addition, it identifies standard properties and value formats. Consistency, especially in the naming scheme used for objects, provides several benefits:

- **A consistent naming scheme provides a guideline for network administrators who will add, modify, or move objects within the Directory tree.**
- **A naming standard eliminates redundant planning and gives network administrators an efficient model to meet their needs, but it leaves implementation of resource objects open and flexible.**
- **Consistent naming schemes help users identify resources quickly, which maximizes user productivity.**
- **Consistent naming enables users to identify themselves easily during login.**

There are two main types of context: current context and object context. Check it out.

Context

Current context is sometimes referred to as *name context*. It defines where you are in the eDirectory tree at any given time, not where you live. This is an important distinction. For example, if you are using a NetWare 6 utility, it's important to know what the utility considers as the current context in the eDirectory tree (that is, the default container to use if one is not specified). This concept is somewhat similar to knowing your current default drive/directory when using a DOS or Windows utility on your workstation.

In addition, current context affects how much of an object's distinguished name you must provide to find it. (See the section "Distinguished Names" later in this chapter for more information.) Current context also enables you to refer to an object in your current container by its common name because the object's context is the same. Note that current context always points to a container object, rather than to a leaf object. Typically, at login, you'll want a workstation's current context set to the container that holds the user's most requently used resources.

In Figure 3.10, Fred's context is ". . . in the ADMIN division of ACME." This context identifies where Fred lives in the eDirectory tree structure. It identifies all container objects leading from him to the Tree Root. In addition to context, Figure 3.10 identifies Fred's common name (CN). A leaf object's common name specifically identifies it within a given container. In this example, the User object's common name is Fred.

Two objects in the same eDirectory tree may have the same common name—provided, however, that they have different contexts. This is why naming is so important. As you can see in Figure 3.11, our eDirectory tree has two Freds, but each has a different context.

Object context (sometimes referred to as *context*) defines where a particular object is located in the eDirectory tree structure. It is a list of container objects leading from the object to the Tree Root. Locating an object through context is similar to locating a file using the directory path. As we learned earlier, object context is used for two important purposes: logging in and accessing resources. Unfortunately, eDirectory does not have a search path feature (such as NetWare SEARCH drives or the DOS PATH command used in the file system). This means that when you request a particular network resource, you (or your workstation) must provide eDirectory with enough information to locate the object in the tree.

Each eDirectory object has a *naming type* (also known as an *attribute type*) associated with it, which allows you to precisely identify objects in your

tree. The tree organization dictates how the objects can be located and used. This naming type is identified by a one- or two-character abbreviation. Accompanying the naming type is the *value* of the object, or the name you enter for the object when you create it. Following are examples of naming types and associated values:

- *C* = Country container
- *O* = Organization container
- *OU* = Organizational Unit container
- *CN* = Common name (specifies a leaf object)

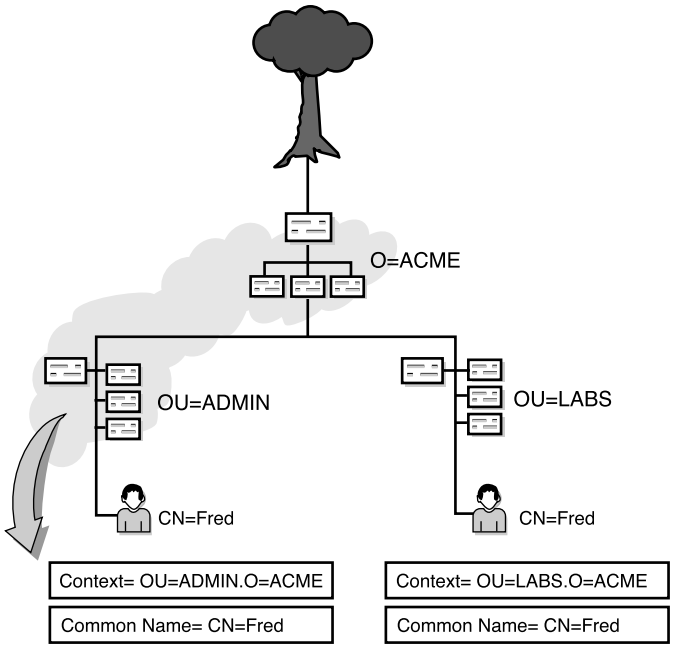

FIGURE 3.11
Understanding eDirectory context.

The Common Name (CN) attribute is the name shown next to the leaf object in the eDirectory tree. This attribute applies to all leaf objects (servers, users, groups, and so on). When requesting a resource such as a server, the Common Name must be included in the request.

Now that you understand how eDirectory context works, review the naming rules associated with it:

- Current context is a pointer to the eDirectory container that your Novell Client is currently set to.
- An object's context defines its location in the eDirectory tree.
- Each object has an identifier abbreviation that defines it for naming purposes, namely the following: C = Country, O = Organization, OU = Organizational Unit, and CN = common name (of leaf object).
- Context is defined by listing all containers from the object to the Tree Root, in that order. Each object is separated by a period.
- Context is important for logging in and accessing eDirectory resources.

There you have it. That's how context works. With this in mind, it's time to explore the two main types of eDirectory names: Distinguished Names (DN) and Relative Distinguished Names (RDN).

Distinguished Names

An object's *distinguished name* is its complete eDirectory path. It is a combination of common name and object context. Each object in the eDirectory tree has a distinguished name that uniquely identifies it in the tree. In other words, two objects cannot have the same distinguished name.

In Figure 3.12, AEinstein's context is .OU=R&D.OU=LABS.O=ACME, and his common name is CN=AEinstein. Therefore, Einstein's distinguished name is a simple mathematical addition of the two:

```
.CN=AEinstein.OU=R&D.OU=LABS.O=ACME
```

Notice the use of periods. A distinguished name always starts with a leading period. Trailing periods aren't allowed. The leading period identifies the name as distinguished (that is, complete). Otherwise, it is assumed to be incomplete. In other words, a relative distinguished name.

Relative Distinguished Names

A *relative distinguished name* lists an object's path to the current context, not the Tree Root. The relativity part refers to how eDirectory builds the distinguished name when you supply a *relative* name. By definition, for example, the common name of a leaf object is a relative distinguished name. When you use a relative distinguished name, eDirectory builds a distinguished name by appending the current context to the end:

Relative distinguished name + current context = distinguished name

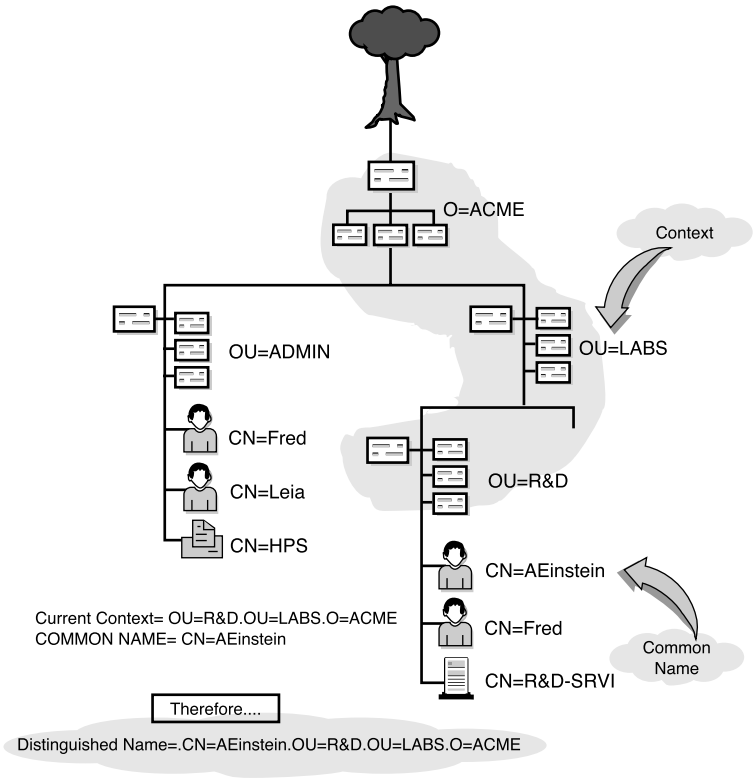

FIGURE 3.12
Building AEinstein's distinguished name.

For example, if the current context is .OU=LABS.O=ACME and you submit a relative distinguished name of CN=AEinstein.OU=R&D, the distinguished name would be resolved as (see Figure 3.13) the following:

```
.CN=AEinstein.OU=R&D.OU=LABS.O=ACME
```

To distinguish a relative name, you must not lead with a period. Instead, you can use trailing periods to change the current context used to resolve the name (as if naming wasn't hard enough already). The bottom line is that each trailing period tells eDirectory to remove one object name from the left side of the current context being used. This concept is somewhat similar to the trailing dot feature used in the DOS CD command.

For example, assume that .OU=R&D.OU=LABS.O=ACME is your current context and that CN=LEIA.OU=ADMIN.. is your relative distinguished name. In this case, the distinguished name would resolve as follows (see Figure 3.14):

```
.CN=LEIA.OU=ADMIN.O=ACME
```

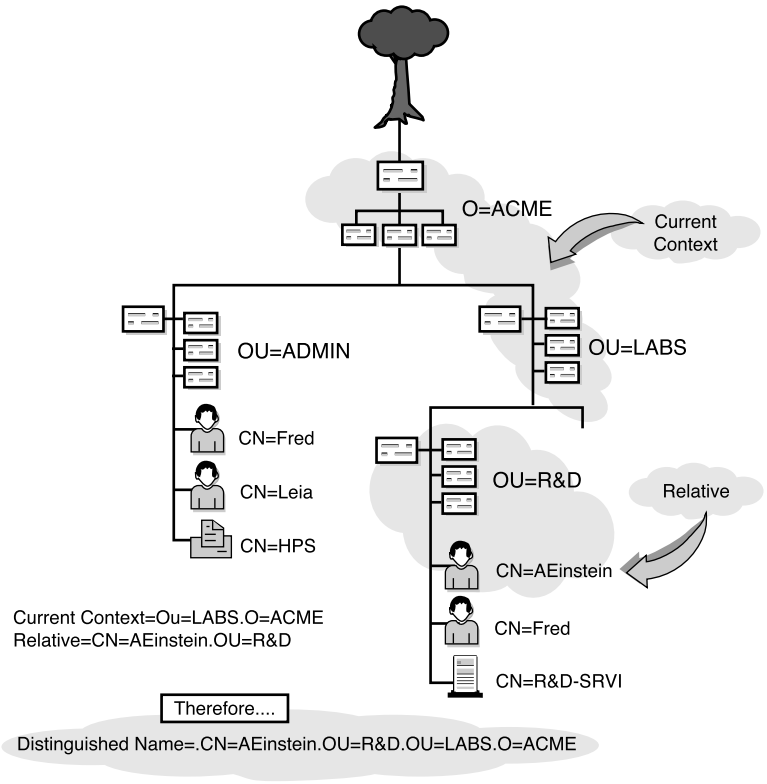

FIGURE 3.13
Building AEinstein's relative distinguished name.

As you can see, it's very important where you place your dots! Here's a quick summary:

- All objects in an eDirectory name are separated by dots.
- Distinguished names are preceded by a dot. This identifies them as complete.
- Relative distinguished names are not preceded by a dot. This identifies them as incomplete.
- Trailing dots can be used only in relative distinguished names because they modify the current context to be used. Each dot moves the context up one container as the distinguished name is resolved.

For a complete summary of eDirectory distinguished naming rules, refer to Table 3.4.

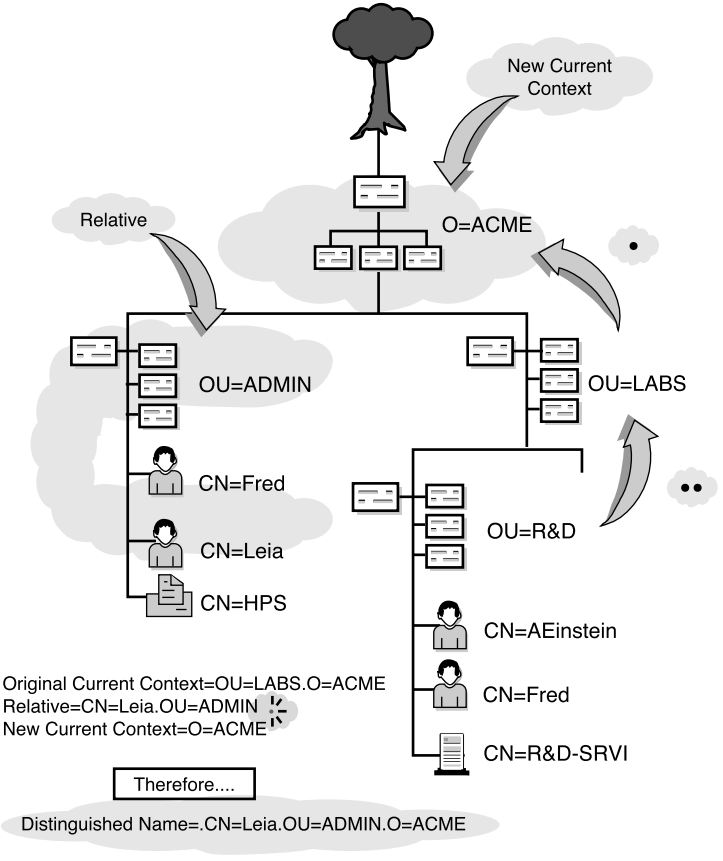

FIGURE 3.14
Using trailing periods to resolve Leia's distinguished name.

TABLE 3.4 **Getting to Know Distinguished Naming**

	DISTINGUISHED NAMES	RELATIVE NAMES
What it is	Complete unique name	Incomplete name based on current context
How it works	Lists the complete path from the object to the Tree Root	Lists the relative path from the object to the current context
Abbreviation	DN	RDN
Leading period	Leading periods required	No leading periods allowed
Trailing periods	No trailing periods allowed	Trailing periods optional

Now, let's step back into reality for a moment and explore the other eDirectory naming category—typeful names.

Typeful Versus Typeless Names

Typeful names use attribute type abbreviations to distinguish between the different container types and leaf objects in eDirectory names. All the examples to this point used these abbreviations to help clarify context, distinguished names, and relative distinguished names. Following are the most popular attribute type abbreviations:

- C = Country container
- O = Organization container
- OU = Organizational Unit container
- CN = Common name of a leaf object

These attribute types help avoid the confusion that can occur when you're creating complex distinguished and relative distinguished names. I highly recommend that you use them. Of course, like most things in life, they are optional. You can imagine how crazy eDirectory naming gets when you choose not to use these attribute abbreviations. This insanity is known as *typeless naming*.

Typeless names operate the same as typeful names do, but they don't include object attribute types. In such cases, eDirectory has to guess what object types you're using. Take the following typeless name, for example:

```
.Admin.ACME
```

Is this the ADMIN Organizational Unit under ACME? Or is this the Admin user under ACME? In both cases, it's a valid distinguished name, except that one identifies an Organizational Unit container, and the other identifies a User leaf object (see Figure 3.15).

Well, here's the bottom line: which one is it? It's up to eDirectory. If you do not provide a typeful object name, eDirectory calculates attribute types for each object. Fortunately, NetWare 6 has some guidelines for guessing what the object type should be:

- The leftmost object is a common name (leaf object).
- The rightmost object is an Organization (container object).
- All middle objects are Organizational Units (container objects).

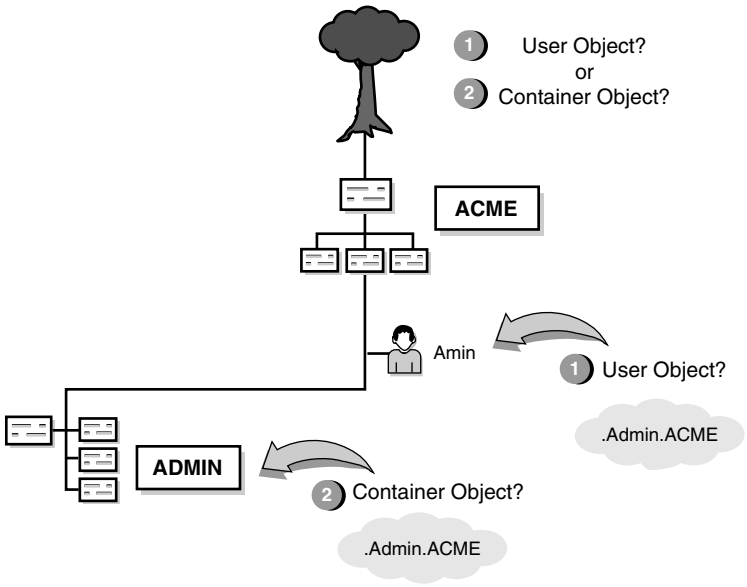

FIGURE 3.15
Trying to understand typeless naming.

Although this works for most cases, it's only a general guideline. Many times, typeless names are more complex. Take the example in Figure 3.15, for instance. You know now that the rightmost object is an Organization, but what about Admin? Is it a common name or an Organizational Unit? We still don't know. Fortunately, NetWare 6 includes a few exceptions to deal with complex typeless scenarios. Here's how it works:

- *Exception Rule 1: Container Objects*—Many NetWare 6 utilities are intelligent enough to resolve typeless names, depending on what they are trying to accomplish. CX, for example, is used primarily for changing context. If you apply the CX command to a typeless name, it assumes the leftmost object is an Organization or Organizational Unit. This is because you can't change your current context to a leaf object. ConsoleOne is the best graphical utility for changing your context. In summary, here's how the example from Figure 3.15 would look with the CX utility:

  ```
  CX .ADMIN.ACME resolves as ".OU=ADMIN.O=ACME"
  ```

- *Exception Rule 2: Leaf Objects*—Similarly, resource-based utilities recognize the leftmost object of a typeless name as a leaf object. Many of these utilities are expecting to see a common name. The most

prevalent are LOGIN, MAP, and CAPTURE. Here's how it works for the example in Figure 3.15:

```
LOGIN .Admin.ACME resolves as ".CN=Admin.O=ACME"
```

- *Exception Rule 3: Contextless Login*—If you have Catalog Services and Contextless Login activated, eDirectory will resolve typeless names by offering the user a list from the eDirectory Catalog. (Note: NetWare 6 eDirectory does not support Catalog Services.)

There you have it. This completes the discussion of typeless names and eDirectory naming in general. As you can see, this is an important topic because it impacts all aspects of eDirectory design, installation, and management. No matter what you do, you're going to have to use the correct name to login to the tree or access eDirectory resources. As you've learned, an object's name is a combination of *what it is* (common name) and *where it lives* (context).

TIP

As a world-class NetWare 6 CNA, you should always use typeful distinguished names in login scripts and capture statements. It is good form, and more importantly, some utilities still require naming attributes for administrative management.

Now you can complete your eDirectory adventure with a quick lesson in changing your current context.

Changing Your Current Context

A user's current context can be set in one of the following ways:

- Before login, using the Name Context field on the Client tab of the Novell NetWare Client Properties (or Novell Client Configuration) window
- During login, using the Context field on the eDirectory tab of the Novell Login window (which is accessed by clicking the Advanced button)
- Using the CONTEXT login script command
- Using the CX utility

It's best to set a user's context at login, so the user can have easy access to the network resources he or she uses the most. If a user wants to access resources located in a different context, he or she will need to use correct

naming conventions. The next sections explore each of these four methods for changing a user's current context before, during, and after login.

Setting a User's Context Before Login

On Windows 95/98 and Windows NT/2000 workstations, you can set the workstation's current context before login by entering the appropriate context information in the Novell NetWare Client Properties window, as shown in Figure 3.16. The typeless distinguished name in the Name Context field sets the workstation's current context before login. It can be entered with or without a preceding period.

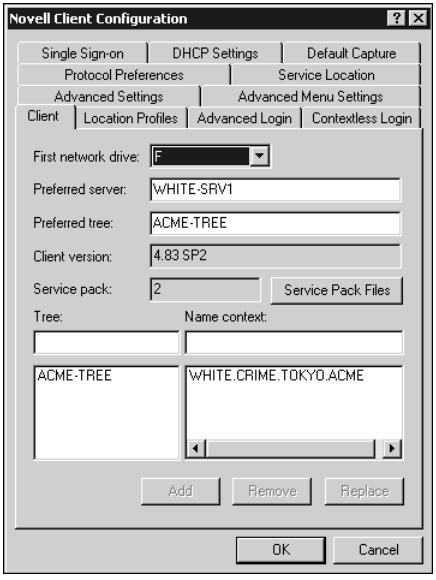

FIGURE 3.16
The Novell NetWare Client Properties dialog box.

To set up a workstation's current context before login, access the Network icon in the Windows 95/98 or Windows NT/2000 Control Panel, select the Novell Client for Windows, and then click Properties. When the Novell Client Properties dialog box appears, enter the current context in the Name Context field on the Client tab. If you're already logged in to the network, a faster way to change this field is to right-click the *N* icon in the system tray and to select Novell Client Properties from the pop-up menu. When the Novell Client Configuration dialog box appears, enter the current context into the Name Context field on the Client tab. (Refer to Chapter 4, "NetWare 6 Connectivity," for more information regarding Novell Client installation and configuration.)

Setting a User's Context During Login

On Windows 95/98 and Windows NT/2000 workstations, you can set a workstation's current context during login by entering the appropriate context information in the Novell Login window. To do so, when the Novell Login window is displayed, click the Advanced button and then enter the current context into the Context field on the NDS tab.

Using the CONTEXT Login Script Command

Setting a workstation's current context during login sets the current context that will be in effect for the user after the user attaches to the network. This prevents the user from having to use distinguished names to access eDirectory resources. Remember, eDirectory attempts to resolve relative distinguished names into distinguished names by appending the current context to the end of the relative name.

The CONTEXT login script command is similar to the NetWare 6 CX command-line utility, except that it does not support all the same options (that is, it sets only the current context). To set a workstation's current context during login, add this command to the appropriate Container, Profile, or User login script:

```
CONTEXT distinguished name
```

For example,

```
CONTEXT .OU=LABS.OU=NORAD.O=ACME
```

or

```
CONTEXT .LABS.NORAD.ACME
```

Note the use of a preceding period (.) to identify the distinguished name. This method is not workstation specific; it can be set for an individual or a group.

Using the CX Command

You can view information about an object's context or change your workstation's current context using the CX command-line utility (which is executed at the DOS prompt on a client workstation). CX is the key NetWare 6 utility for dealing with eDirectory context. It enables you to perform two important tasks: change your workstation's current context and/or view information about a resource's object context.

CX is a relatively straightforward command with a great deal of versatility. In fact, it's similar to the DOS CD command in its general approach. If you type **CX** by itself, the system displays your workstation's current context. This is marginally interesting, at best. CX really excels when you combine it with one or more command-line switches. Following are some of the more interesting ones:

- *CX*—View your workstation's current context.
- *CX .*—Move the context up one container for each period (.). Don't forget the space between CX and the period (.).
- *CX /T*—View the Directory tree structure below your current context.
- *CX /A /T*—View all objects in the Directory tree structure below your current context.
- *CX /R /A /T*—View all objects in the Directory tree below the Tree Root.
- *CX /CONT*—List containers only, below the current context, in a vertical list, with no directory structure.
- *CX /C*—Scroll output continuously.
- *CX .OU=ADMIN.O=ACME*—Change your current context to the ADMIN container of ACME.
- *CX /?*—View online help, including various CX options.
- *CX /VER*—View the version number of the CX utility and the list of files it executes.

Probably the most useful CX option is CX /R/A/T. I'm sure there's a hidden meaning there somewhere in the rodent reference. Or, if you lean toward the artistic side of the fence, try CX /A/R/T. And finally, for all you urbanites, there is CX /T/A/R. Regardless, the CX /R/A/T option displays the relative location of all objects in the eDirectory tree.

Understanding Inheritance

eDirectory rights can also be obtained through inheritance. In simple terms, inheritance occurs when rights granted to a trustee of a container flow down to all objects within and below the container. Inheritance minimizes the individual rights assignments needed to administer the network because object/property rights can automatically flow down the tree from containers to subcontainers to leaf objects. Inheritance is an automatic side effect of trustee assignments. Both object and property rights can be inherited.

Congratulations! This completes the lesson on eDirectory.

So far, you've explored the basics of NetWare 6 and the intricacies of eDirectory. Now you're ready for Prime Time:

- NetWare 6 Connectivity (Chapter 4)
- NetWare 6 File System (Chapter 5)
- NetWare 6 Security (Chapter 6)
- NetWare 6 Advanced Security (Chapter 7)
- NetWare 6 Queue-Based Printing (Chapter 8)
- NetWare 6 NDPS Printing (Chapter 9)
- NetWare 6 Messaging Services (Chapter 10)
- NetWare 6 Internet Infrastructure (Chapter 11)

Don't be scared—I'll be with you every step of the way. And we'll explore eDirectory many times throughout this CNA Study Guide.

Well, that's all there is to it — not really! Remember, the glass is half full, and you have taken a huge "gulp" in this chapter. Use this information wisely as you expand the horizons of your network. Next, the discussion of NetWare 6 administration features continues as you learn how to connect to this mysterious and exciting network Directory.

See ya there!

Lab Exercise 3.2: Understanding eDirectory Naming

Answer the following questions using the directory structure shown in Figure 3.17.

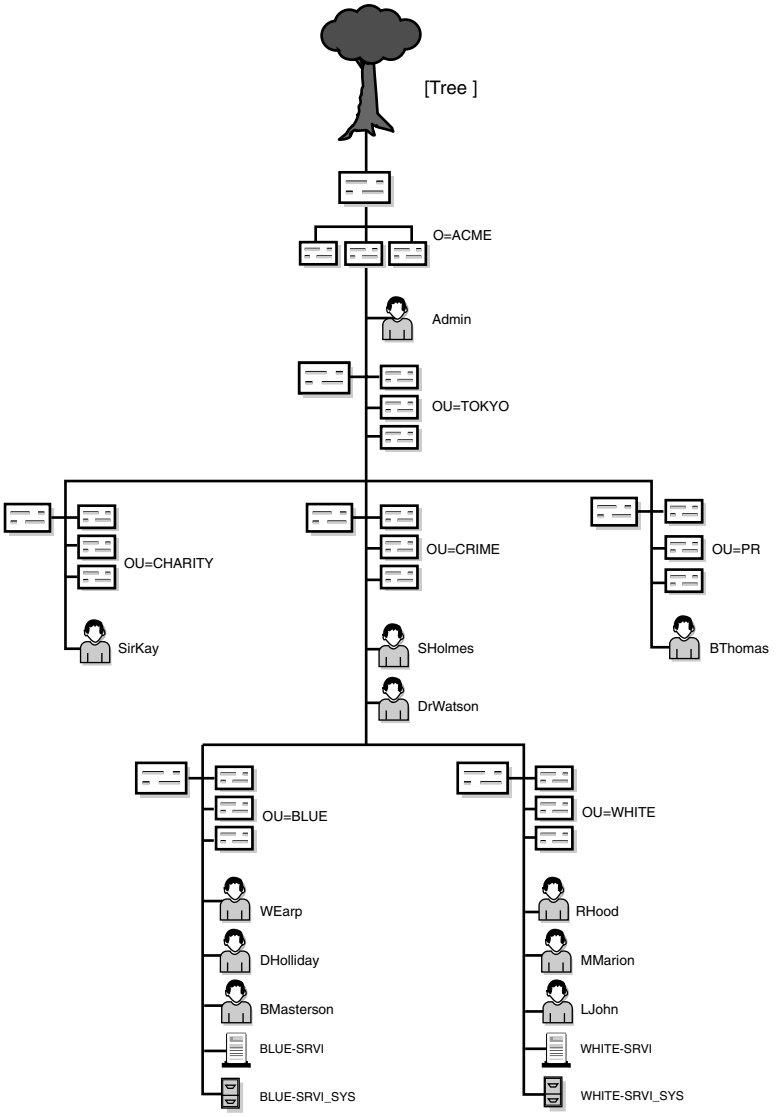

FIGURE 3.17 Understanding eDirectory naming for Tokyo.

1. Indicate a typeless distinguished name for BMasterson.
2. Provide a typeful distinguished name for RHood.
3. List a typeless relative distinguished name for the CRIME Organizational Unit, assuming that your current context is the Tree Root.
4. Show a typeful relative distinguished name for the BLUE-SRV1 Server object from the default current context.
5. If your current context is .CRIME.TOKYO.ACME, what is the shortest name that accurately references the SHolmes User object?
6. Assume your current context is .TOKYO.ACME. Indicate a typeless relative distinguished name for the LJohn User object.
7. If your current context is .PR.TOKYO.ACME, what would be a typeful relative distinguished name for SirKay?
8. Assume your current context is .WHITE.CRIME.TOKYO.ACME. Provide a typeless relative distinguished name for Admin.
9. If your current context is .BLUE.CRIME.TOKYO.ACME, what would be a typeful relative distinguished name for BThomas?
10. Assume your current context is .WHITE.CRIME.TOKYO.ACME. What is the longest possible typeful relative distinguished name for the SYS: volume on the BLUE-SRV1 server?
11. If DHolliday attaches to the BLUE-SRV1 server by default, what is his current context after login? Give two LOGIN commands for DHolliday.
12. How would MMarion visit SirKay?
13. Provide 10 LOGIN commands for SHolmes from .BLUE.CRIME.TOKYO.ACME.
14. What is the easiest way to move above ACME from the .PR.TOKYO.ACME context?

See Appendix C for answers.