An Introduction to NetWare Directory Services

Articles and Tips: article

GAMAL B. HERBON
Senior Technical Editor
Systems Research

01 Apr 1993

NetWare Directory Services (NDS) is a globally distributed network database that replaces the bindery used in previous versions of NetWare. This AppNote introduces the basic concepts behind NDS, discussing NDS objects and properties and telling how Root, Container, and Leaf objects form the Directory tree. t also explains about NDS partitions and replicas, bindery compatibility, and time synchronization.

Introduction

The most noticeable new feature of NetWare 4.0 is its globally distributed Directory. NetWare Directory Services (NDS) is more than just a global naming service; it also provides an easily managed and more secure network environment. With NDS, it is now possible to integrate a diverse network of resources into a single, easy-to-use environment.

Note The term "Directory" (with a capital D) refers specifically to the global, distributed database provided in NetWare 4.0. The NDS Directory is different from the file system directory and its structure.

This AppNote introduces the basic concepts behind NetWare Directory Services. After giving a quick overview of NDS, it discusses NDS objects and properties and explains how these objects form the Directory tree. t also explains about NDS partitions and replicas, bindery compatibility, and time synchronization.

The next AppNote in this issue builds on these concepts and provides guidelines for planning your own Directory tree. It also gives detailed examples of setting up a small, medium-sized, and large Directory tree.

Overview of NetWare Directory Services

NetWare Directory Services is a global, distributed, and replicated database. As part of NetWare 4 0, NDS maintains information about all network resources (users, groups, servers, volumes, printers, and so on) in a hierarchical tree structure. Network resources can be organized in the tree independent of their physical location. Thus network users can access any network resource they have rights to, without having to know the exact location of that resource.

The Directory replaces the NetWare bindery, which served as the system database for previous versions of NetWare. Rather than supporting a single server (as the bindery did), NDS supports an entire network of servers. Distributing the network database allows all servers to easily access all network information. It also allows the database to be replicated, thus minimizing the risk of a single point of failure.

NetWare 4.0 provides compatibility with bindery-based versions of NetWare through the bindery emulation feature of NDS.

NDS is based on parts of the CCITT X.500 standard. By not locking NDS strictly into this proposed standard, Novell allows for future expansion of the Directory and of the possible services it can provide.

Network-Wide Login

With NDS, users no longer need to login or attach to specific servers. Instead, they can login to the *network*. For example, a user could log in to the network by typing:

```
LOGIN GHERBON
```

instead of

```
LOGIN servername/GHERBON
```

Once logged in to the network, users can access any service or resource they have rights to, without having to explicitly login or attach to other servers. The users will be transparently attached to the server on which the specified service resides. NDS handles all of the address resolution issues in the background, so users are shielded from the complexity of having to

Authentication

In NetWare 4.0, users' access to network resources is restricted by the rights they are assigned in the Directory. When a user accesses network resources (such as servers, volumes, and printers), authentication occurs in the background. The authentication process verifies that the user has sufficient rights to use the requested resource.

The network-wide login and background authentication that NDS provides effectively locks out unauthorized users and makes using the network and accessing resources easier for authorized users. Users need only one password to gain access to all network resources available to them. Of course, the available resources will be limited to those the user has been granted rights to.

A New Mindset

For years, NetWare has relied on the bindery to store and provide all information necessary for the operating system and applications. The bindery contained information about users and groups, valid passwords, rights, attached printers, and other network resources. However, adding new users to a server (or especially to several servers) was a tedious, time-consuming process.

If you use NDS with NetWare 4.0, you have at your disposal a powerful yet easy-to-use computing environment, complemented by enhanced security and network management capabilities. However, before you can simplify network administration through the use of NDS, you need to adjust to a completely new mindset. You need to view the network as a unified information system rather than a fragmented collection of computers. This new mindset will take some getting used to, but it will make enterprise networking feasible and desirable - even among those from the mainframe world who have argued against the use of PC-based networks.

NDS provides for easier management of the network resources listed in the Directory. However, it is important to remember that the Directory does not directly control the NetWare file system (volumes, directories, and files). NetWare 4.0 provides text-based and graphical utilities to manage both NDS and the file system.

NDS Objects

The NDS Directory tree is formed by placing "objects" in a hierarchical tree structure. NDS objects consist of categories of information, known as *properties*, and the data included in those properties. This information is stored in the Directory database.

The NDS database can contain three types of objects:

- Physical objects (such as users and printers)
- Logical objects (such as groups and print queues)
- Other objects (such as Organizational Units) designed to help organize and manage the physical and logical objects

It is important to understand that NDS objects are structures that store information, not the actual entity represented by the object. For example, a Printer object stores information about a specific printer and helps manage how the printer is used, but it is not the physical printer itself.

Object Properties

As mentioned above, properties are categories of information stored in the database for NDS objects. Each NDS object has properties that contain information about that object. For example, this information may include a user's telephone number and physical address, or the physical location of a printer.

You enter the information, or *values*, about the object into data fields for each property. For example, a User object includes the following properties:

- Login Name
- Telephone Number
- E-mail Address
- Password Restrictions
- Group Membership
- Address
- And others ...

Figure 1 shows the relationship between objects, properties, and values.

Figure 1: An NDS object (such as a user) consists of numerous properties with corresponding values.

Object	*Property*	*Value*
User	Login NameEMailAddressTelephone NumberAddress	GHerbonGHerbon@Novell800-555-4321123Oak StreetAnywhere, USA xxxx

In many cases, you can enter more than one value for a property. An example is the Telephone Number property for User objects. In this property, you can enter values for a user's office phone number, home phone number, cellular phone number, and pager number.

Once the values are entered in the object properties, you can perform a search for objects with specific values. For example, if you request information that specifies a certain area code, the Directory database could return all telephone numbers that contained the specified area code in their properties.

You can also request information on a specific object and receive information on all properties of that object which you have access to.

- File system file rights
- NDS object rights
- NDS property rights

Previous versions of NetWare had file system directory and file rights, and very limited "access levels" to bindery objects. NetWare 4.0 adds NDS object and NDS property rights, which determine what you can do within the Directory. For brevity, we'll simply call these object and property rights. Since this AppNote deals only with NDS, we won't discuss file system rights here.

The concepts about NDS object and property rights summarized here are discussed in greater detail in the "Understanding Directory Services Rights" AppNote in this issue.

Because the Directory is a hierarchical tree structure, rights assigned in the Directory flow down through the tree. This is an important concept to understand when you are designing your Directory tree.

To provide better access control to the pieces of information (properties) contained in NDS objects, object and property rights are assigned separately.

- *Object rights* control what a trustee is allowed to do with the object. These rights include Browse, Create, Delete, Rename, and Supervisor.

  Object rights control access to an NDS object as a single piece of the Directory tree, but they do not allow access to information stored within that object (its properties). The only exception is the Supervisor object right, which applies to an object's properties as well as to the object itself.

- *Property rights* control a trustee's access to information associated with the object (in the object's properties). These rights include Compare, Read, Write, Add or Delete Self, and Supervisor.

  Property rights apply only to NDS object properties, not to the objects themselves. For example, if you include a telephone number as a property fora User object, you can prevent anyone else from seeing the specified telephone number by not granting them the Read right to that particular property. At the same time, you can still allow the person to view other properties, such as the user's address. This allows flexibility in deciding what information others can access.

Access Control List. The information about who can access object information is stored in the object itself, in a property known as the Access Control List (ACL). The ACL property contains the trustee assignments and the Inherited Rights Filter (explained below).

An object's ACL defines which objects can access that object and its properties. For example, an object listed in a Printer object's ACL is a trustee of that Printer object and therefore has some rights to the printer. To change the trustee's access to the Printer object, you would change the trustee's entry in the Printer object's ACL. Only trustees with the Write right for the ACL property can change the trustee assignments or the Inherited Rights Filter.

Each object listed in an ACL can have different rights to that object's properties. For example, if ten users are listed in the Modem object's ACL as trustees, each of those ten users can have different rights to that Modem object and to its properties. However, in actual use, it is likely that at least some of the users will have the same rights (or at least similar rights) to the Modem and its properties.

Inherited Rights Filter. While trustee assignments grant access to an object, the Inherited Rights Filter (RF) prevents rights from automatically propagating from one object to another. In the Directory tree, an object can automatically receive, or "inherit," rights from its parent objects. The IRF can be used to block any or all of these inherited rights so that no objects can receive them. Every object and property in the Directory can have an Inherited Rights Filter.

Effective Rights. The combination of inherited rights, trustee assignments in an ACL, and security equivalences are known as *effective rights*. An object's effective rights are what control its access to another object and that object's properties.

How Objects Form the Directory Tree

NDS operates in a logical organization called the Directory tree. This is so named because objects are stored in a hierarchical tree structure. By time-honored computer science convention, this structure has the tree growing upside down starting with the [Root] object at the top of the tree and branching downward.

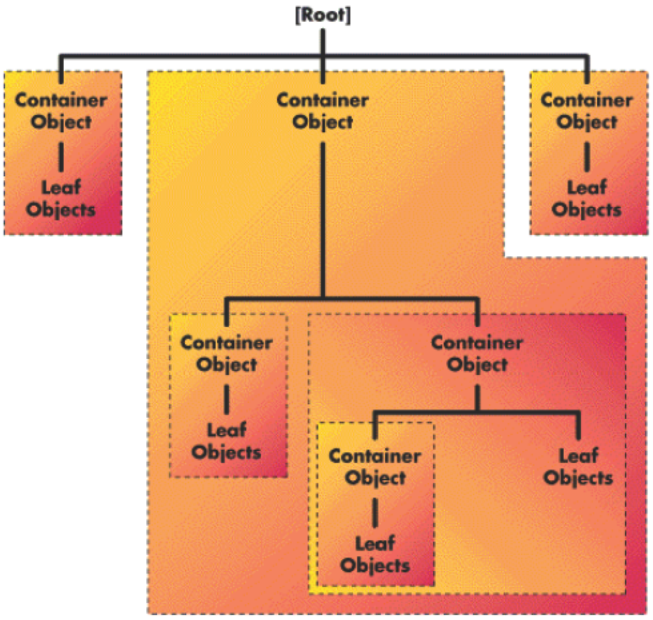
The Directory tree is made up of three types of objects:

- The [Root] object
- Container objects
- Leaf objects

The [Root] object is automatically placed at the top of the tree by the NetWare 4.0 installation program. Branches of the Directory tree consist of container objects and all of the objects they hold. These container objects can also contain other container objects. Leaf objects are at the ends of the branches and do not contain any other objects.

Figure 2 illustrates how objects can be laid out to form the Directory tree.

Figure 2: The Directory tree is formed by container objects and leaf objects branching down from the [Root] object.



The sections which follow discuss the three types of objects in greater detail.

[Root] Object

The [Root] object can only be created by the NetWare 4 0 installation program, which automatically places it at the top of the tree. The [Root] object cannot be renamed or deleted.

Note The [Root] object of a Directory tree should not be confused with the root directory in the file system. In the file system, the root directory is the first directory on a volume. It bears no relation to the [Root] object of a Directory tree.

The [Root] object can have trustees, and the [Root] object trustees' rights flow down the tree. One example is the User object Admin, which is created automatically during installation. By default, Admin receives a trustee assignment of Supervisor rights to the [Root] object of the Directory tree. This gives Admin all rights to all objects and properties in the tree, so that it can be used to initially log in and set up the tree.

(For more information about Admin, refer to the "User Object Admin" section later in this AppNote.)

The [Root] object can also be a trustee. However, you should give careful consideration before making [Root] a trustee of another object. If you do, *every* object in the tree has the same rights as the [Root] object by virtue of ancestor inheritance. In effect, you will have made *all* users security equivalent to [Root].

Container Objects

Container objects hold (or contain) other Directory objects. Container objects are provided as a means of logically organizing all other objects in the Directory tree. Just as directories are used to group related files together in a file system, container objects are used to group related items in the Directory tree.

There are two kinds of container objects: Organization and Organizational Unit. These are explained below.

Note The NDS Directory also supports Country and Locality as container objects. In normal usage, however, these container objects should not be necessary as they can add unnecessary complexity to your Directory tree. For this reason, we do not cover these options.

Organization (O). An Organization object helps you organize other objects in the Directory tree. It also allows you to set defaults for User objects you create in the Organization container.

You can use an Organization object to designate a company, a division of a company, a university or college with various departments, or a department with several project teams.

Important Use of the Organization object is mandatory. Every Directory tree must contain at least one Organization object. Organization objects must be placed one level below the [Root] object.

Organizational Unit (OU). An Organizational Unit object helps you to organize leaf objects in the Directory tree. t also allows you to set defaults in a login script, and create a user template for User objects you create in the Organizational Unit container.

You can use an Organizational Unit object to designate a business unit within a company, a department within a division or univeristy, a project team within a department, and so on.

Important Use of Organizational Unit objects is optional in a Directory tree. If used, Organizational Units must be placed one level below an Organization or another Organizational Unit.

In the initial NetWare 4.0 release, you cannot easily change the name of a container object once it is named. To avoid possible problems, you should carefully plan the names of your container objects before implementing your Directory tree.

Leaf Objects

Directory leaf objects are objects that do not contain any other objects. These represent actual network entities such as users, servers printers computers and so on. The sections below list and describe the different types of leaf objects available in

which includes default rights assignments. You can also define a USER_TEMPLATE object to provide new users with default settings that you have already decided on.

Users with NetWare 4.0 workstations (those who use NDS rather than bindery emulation) can be created anywhere in the Directory tree. They must know their exact NDS context in order to log in. To make this easier, enter users' context in their workstation NET.CFG file when you install their NetWare 4 0 workstations. This setting automatically places them in the correct context every time they login from their workstation.

Users with non-4.0 workstations must be created in the container where the bindery emulation context is set for their primary server. Remember that bindery emulation is set (by default) for every NetWare 4 0 server that is installed. Non- 4 0 users do not need to know their context because they are logging in to a server rather than the Directory tree. *(For more information, see the "Bindery Compatibility" section of this AppNote.)*

Group

A Group object assigns a name to a list of User objects located anywhere in the Directory tree. Use a Group object when you want to assign rights to a group as a whole, rather than just individual users. The rights assigned to a Group object are granted to individual users who are members of the group, no matter where they are located in the Directory tree.

Profile

A Profile object contains a profile script (a type of login script). The Profile object listed as a property in a User object is executed when that User objects logs in to the network. The Profile object is executed after the system login script, but before the user login script.

Create a Profile object for any set of users who need to share common login script commands, but who are not located in the same Directory container, or for any users who are a subset of users in the same container.

Organizational Role

An Organizational Role object defines a position or role within an organization. An example might be a department manager or vice president of sales, and so on. You can assign any User object to be an occupant of the Organizational Role object. Any occupant receives the same rights that were granted to the Organizational Role object.

You create an Organizational Role object to assign rights to a particular position in the organization where the person holding the position might change frequently, while the actual responsibilities of the position do not change often. t can also be used when you have a job where you want different people to handle the same job at different times of the year.

For example, suppose you wanted a Print Manager for the Sales department, but you do not want the same person to do the job for more than a one- month period. You could create an Organizational Role object called PRINT MANAGER and grant that object all object rights to the Printer, Print Queue, and Print Server objects in that part of the Directory tree. You might also grant the PRINT MANAGER object the property rights to the Print Job Configuration property of users. This allows the PRINT MANAGER Organizational Role object to manage all printing in the SALES container, without having to grant these rights to individual users.

Server-Related Leaf Objects. The following leaf objects are related to NetWare servers and volumes.

NetWare Server

A NetWare Server object represents a server running NetWare on your network. Whenever you install a server in the tree, a NetWare Server object is automatically created.

Use this object to store information about the server in the NetWare Server object's properties. This can include such information as the server's location on the wire, the server's physical location, what services the server provides, and so on.

In addition to storing information about the NetWare server, this object affects the network in that it is referred to by several other objects in the Directory. One example is the NDS Volume object, which points to the NetWare Server object to find a physical volume on the network. Another example is the Directory Map object, which points to the NetWare Server object to find the file system directory it needs.

The NetWare Server object is also used to tie the physical server on the network to the Directory tree. Without this object you cannot access file systems that reside on the server's volumes.

For informational purposes, you can also create NetWare Server objects for servers *not* in the Directory tree (such as 3.11 servers not in the tree).

Volume

A Volume object represents a physical volume on the network. INSTALL automatically creates a Volume object for every physical volume on a server at installation time.

In the Volume object's properties, you store information about which NetWare server the physical volume is located on, and the name given when the volume was initialized at installation (such as SYS). If you create a Volume object during installation, this necessary information is placed in the Volume object's properties by default.

Properties in the Volume object are also used for mapping drives.

In the NetWare Administrator (GUI utility version), you can click on the Volume object icon to display information about the file system directories and files located on that volume.

Directory Map

A Directory Map object represents a particular directory path or file in the file system of a given server. This is currently used only by the MAP utility. Directory Map objects are especially useful in login scripts, because they can be set to point to directories that contain applications or other frequently used files.

For example, if you have a directory that contains DR DOS 6.0, you will probably map a search drive to that directory in all login scripts you create. If you later decide to upgrade to a newer version of DR DOS and rename the directory, you have to change the mapping in every login script that contains that search mapping.

By using the Directory Map object, you avoid the necessity of making all these login script changes. Instead, you just change the Directory Map object, and all the search mappings in your login scripts are updated to find the new version automatically.

Printer-Related Leaf Objects. The following leaf objects are related to NetWare's print services. These objects are created and controlled using the NetWare print utilities.

Print Queue

A Print Queue object represents a print queue on the network. You must create a Print Queue object for every print queue on

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.