Novell
eDirectory™

Novell®

Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

This product may require export authorization from the U.S. Department of Commerce prior to exporting from the U.S. or Canada.

Copyright © 1993-2001 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

U.S. Patent No. 5,608,903; 5,671,414; 5,677,851; 5,758,344; 5,784,560; 5,794,232; 5,818,936; 5,832,275; 5,832,483; 5,832,487; 5,870,739; 5,873,079; 5,878,415; 5,884,304; 5,913,025; 5,919,257; 5,933,826. U.S and Foreign Patents Pending.

Novell, Inc.
1800 South Novell Place
Provo, UT 84606
U.S.A.

www.novell.com

Novell eDirectory Administration Guide
October 2001
103-000155-001

Online Documentation: To access the online documentation for this and other Novell products, and to get updates, see www.novell.com/documentation.

Novell Trademarks

BorderManager is a trademark of Novell, Inc.

ConsoleOne is a trademark of Novell, Inc.

DirXML is a trademark of Novell, Inc.

eDirectory is a trademark of Novell, Inc.

IPX is a trademark of Novell, Inc.

ManageWise is a trademark of Novell, Inc.

NDS Manager is a trademark of Novell, Inc.

NetWare is a registered trademark of Novell, Inc. in the United States and other countries.

Novell is a registered trademark of Novell, Inc. in the United States and other countries.

Novell Certificate Server is a trademark of Novell, Inc.

Novell Client is a trademark of Novell, Inc.

Novell Directory Services and NDS are registered trademarks of Novell, Inc. in the United States and other countries.

Novell Modular Authentication Services and NMAS are trademarks of Novell, Inc.

Novell Support Connection is a registered trademark of Novell, Inc. in the United States and other countries.

Transaction Tracking System and TTS are trademarks of Novell, Inc.

SMS is a trademark of Novell, Inc.

ZENworks is a trademark of Novell, Inc.

Third-Party Trademarks

All third-party trademarks are the property of their respective owners.

Contents

Novell eDirectory

Novell® eDirectory™ is a highly scalable, high performing, secure directory service. It can store and manage millions of objects, such as users, applications, network devices, and data. Novell eDirectory natively supports the directory standard Lightweight Directory Access Protocol (LDAP) 3 over Secure Socket Layer (SSL).

Novell eDirectory includes public key cryptography services that allow you to protect confidential data transmissions over public communication channels such as the Internet.

Novell eDirectory provides the basic foundation for the directory service that provides replication and partitioning capabilities, along with other utilities. Additional products that build upon this basic directory structure from Novell are also available to increase functionality.

This documentation includes the following sections:

Documentation Conventions

In this documentation, a greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path.

Also, a trademark symbol ($^{®}$, ™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

1 Installing and Upgrading Novell eDirectory

Novell® eDirectory™ currently runs on NetWare®, Windows* NT*/2000 Server, Linux*, and Solaris*.

The following sections provide information you will need before installing Novell eDirectory, while you are using the installation program, and information about uninstalling Novell eDirectory from various platforms:

- "Common Installation Concepts" on page 18
- "Installing eDirectory for NetWare" on page 20
- "Installing eDirectory for Windows NT/2000 Server" on page 26
- "Installing Novell eDirectory on Linux or Solaris" on page 29
- "Configuring Novell eDirectory on Linux or Solaris Systems" on page 48
- "Uninstalling eDirectory on NetWare" on page 54
- "Uninstalling eDirectory on Windows NT/2000" on page 54
- "Uninstalling eDirectory from Linux or Solaris Systems" on page 55

If you are new to eDirectory, review Chapter 2, "Designing Your Novell eDirectory Network," on page 57 and Chapter 3, "Understanding Novell eDirectory," on page 85 before installing Novell eDirectory.

Common Installation Concepts

Understanding the following concepts can help you make decisions as you install eDirectory on the various platforms:

- ◆ "Hardware Requirements" on page 18
- ◆ "Forcing the Backlink Process to Run" on page 19

Hardware Requirements

Hardware requirements depend on the specific implementation of eDirectory.

For example, a base installation of eDirectory with the standard schema requires about 74 MB of disk space for every 50,000 users. However, if you add a new set of attributes or completely fill in every existing attribute, the object size grows. These additions require more disk space, processor, and memory needed.

Two factors increase performance: more cache memory and faster processors.

For best results, cache as much of the DIB Set as the hardware allows. See "Distributing Memory between Entry and Block Caches" on page 433.

eDirectory scales well on a single processor. However, eDirectory 8.6 takes advantage of multiple processors. Adding processors improves performance in some areas, for example, logins, and having multiple threads active on multiple processors. eDirectory itself is not processor-intensive, but it is I/O-intensive.

Table 1 illustrates typical system recommendations for eDirectory for NetWare, Windows NT/2000, and Linux.

Table 1   System Requirement for NetWare, Windows, and Linux

Objects	Processor	Memory	Hard Disk
100,000	Pentium* III 450-700 MHz (single)	384 MB	144 MB
1 million	Pentium III 450-700 MHz (dual)	2 GB	1.5 GB
10 million	Pentium III 450-700 MHz (2 to 4)	2 GB +	15 GB

Table 2 illustrates typical system recommendations for eDirectory for Solaris.

Table 2   **System Requirement for Solaris**

Objects	Processor	Memory	Hard Disk
100,000	Sun* Enterprise 220	384 MB	144 MB
1 million	Sun Enterprise 450	2 GB	1.5 GB
10 million	Sun Enterprise 4500 with multiple processors	2 GB +	15 GB

Requirements for processors might be greater than the tables indicate, depending upon additional services available on the computer, as well as the number of authentications, reads, and writes that the computer is handling. Processes such as encryption and indexing can be processor-intensive.

Of course, faster processors improve performance. Additional memory improves performance because eDirectory will then cache more of the directory into memory. Of course, faster processors improve performance.

Forcing the Backlink Process to Run

Because the internal eDirectory identifiers change, when upgrading to Novell eDirectory 8.6, the backlink process has to update backlinked objects for them to be consistent.

Backlinks keep track of external references to objects on other servers. For each external reference on a server, the backlink process ensures that the real object exists in the correct location and verifies all backlink attributes on the master of the replica. By default, the backlink process occurs two hours after the database is open and then every 780 minutes (13 hours). The interval is configurable from 2 minutes to 10,080 minutes (7 days).

After migrating to eDirectory 8.6, we recommend that you force the backlink to run by issuing a **SET DSTRACE=*B** command from the server console. On Linux and Solaris systems, run this command from the ndstrace command prompt. Running the backlink process is especially important on servers that do not contain a replica.

Installing eDirectory for NetWare

eDirectory for NetWare can co-exist with the following NDS® and eDirectory versions:

- NetWare 4.11 or 4.2 with NDS 6.09 or later

- NetWare 5 with Support Pack 5 or later (http://support.novell.com/misc/patlst.htm#nw), and NDS 7.47 or later (but earlier than NDS 8)

- NetWare 5 with Support Pack 5 or later (http://support.novell.com/misc/patlst.htm#nw), and NDS 8.51 or later

- NetWare 5.1

- NDS eDirectory 8.5 on Windows NT/2000, NetWare, Solaris, Linux, or Tru64 UNIX*

- Novell eDirectory 8.6 on Windows NT/2000, NetWare, Solaris, or Linux

If your eDirectory tree does not have a Novell Certificate Server™, the eDirectory installation program does the following:

- Creates a Security container object for the entire eDirectory tree

  This object is created at the top of the eDirectory tree and must remain there.

- Creates an Organizational Certificate Authority (CA) object

- Places the Organizational CA object in the Security container

Only one Organizational CA object can exist in an eDirectory tree. Because you must not move this object from one server to another, ensure that the first eDirectory server is the one that you intend to permanently host the Organizational CA object. For more information, see "Understanding the Novell Certificate Server" on page 75.

System Requirements

❑ If you are using RCONSOLE, you will need a ConsoleOne™ administrator workstation with the following:

- A 200 MHz or faster processor

- A minimum of 64 MB RAM (128 MB recommended)

❑ Novell Client™ software that shipped with NetWare 5 or later.

❑ Administrative rights to the eDirectory tree, so that you can modify the schema.

For additional information, see the following:

- System Requirements for eDirectory in *AppNotes* (http://
developer.novell.com/research/appnotes/2000/july/03/a000703.htm)

- "Hardware Requirements" on page 18

Prerequisites

If you are installing into a tree that has NetWare 5 servers, each NetWare 5 server must be running NetWare 5.0 SP5 Support Pack 6a (http://support.novell.com/misc/patlst.htm#nw) or later.

If you are upgrading eDirectory 8.6 on a NetWare 5.1 server, the NetWare 5.1 server must be running NetWare 5.1 SP2a (http://support.novell.com/misc/patlst.htm#nw) or later.

Updating the eDirectory Schema for NetWare

When upgrading a NetWare 5.*x* server to eDirectory 8.6, you must update the eDirectory schema by running DSREPAIR on the server that has the master replica of the Tree partition.

IMPORTANT: If the master replica of the Tree partition resides on an NT server, follow the instructions in "Updating the eDirectory Schema for NT/2000" on page 27.

If one or both of the following conditions exist, you must run DSREPAIR.NLM before installing the first eDirectory server in your tree:

- Any NetWare 5 server in your eDirectory tree is running eDirectory 8.

- Your first installation of eDirectory is on a NetWare 5 server that does not hold a writeable replica of the root 8.6 partition.

To update the schema:

1 Copy the appropriate DSREPAIR.NLM file from the product CD (or downloaded and expanded file) to the SYS:\SYSTEM directory of the server that contains the master replica of the Tree partition.

Table 3   DSREPAIR Versions

For This Version of NetWare	With This Version of NDS	Copy
4.11 or 4.2	6.09 or later	PATCHES\DSREPAIR\ NW4X\DSREPAIR.NLM
5.0 or later	NDS 7 (version 7.47 or later)	PATCHES\DSREPAIR\ NW5X\DSREPAIR.NLM
5.0 or later	8.11 or 8.17	Not supported
5.0 or later	8.51 or later	PATCHES\DSREPAIR\ NWNDS8\DSREPAIR. NLM

2 At the server console of the master replica of the Tree partition, load DSREPAIR.NLM.

3 Select Advanced Options Menu > Global Schema Operations.

4 Enter the Administrator's name (for example, Admin.VMP) and password.

5 Select Post NetWare 5 Schema Update > Yes.

DSREPAIR.NLM updates the schema and posts the results to the DSREPAIR.LOG file.

Ignore errors associated with adding object classes. DSREPAIR.NLM is simply applying the Post NetWare 5 Schema Update changes to each object.

6 Copy the appropriate patch version of DSREPAIR.NLM to each NetWare server in the eDirectory tree.

Use Table 3 as a reference. Having a correct version on each server ensures that the schema needed for eDirectory is properly maintained when DSREPAIR.NLM is run in the future.

If you use an earlier version of DSREPAIR.NLM and select Rebuild Operational Schema, schema enhancements made by the Post NetWare 5 Schema Update will be lost. To resolve lost schema enhancements, run DSREPAIR.NLM according to the following table.

If you are running DSREPAIR.NLM from here	Do this
A server that holds a writable replica of the root partition	Reapply the Post NetWare 5 Schema Update to your eDirectory tree.
From any other server	Click Advanced Options > Global Schema Operations > Request Schema from Tree.

This action resynchronizes the schema from the root of the tree.

7 Close DSREPAIR.NLM before installing eDirectory on the server.

If DSREPAIR.NLM is loaded, the server might not restart.

Installing a Support Pack

Table 4   Installing a Support Pack

Before installing eDirectory on this server	Install this support pack
NetWare 5	NetWare 5 Support Pack 6a
NetWare 5.1	NetWare 5.1 Support Pack 2a or later

1 (Conditional) Download the latest support pack to the NetWare 5.*x* server.

For example, download to SYS:\.

If you purchased a support pack CD from Support Connection, skip this step.

2 (Conditional) Expand the support pack.

For NW51SP1.EXE, the support pack might take several minutes to verify an ARJ-SECURITY envelope. After verification, the support pack creates an NW5SP*X* directory and places subdirectories and files there.

If you purchased a CD from Support Connection, skip this step. The support pack is already expanded.

3 At the server console, start NWCONFIG.NLM.

4 Select Product Options > Install a Product Not Listed.

5 Press F3 (F4 if you're using RCONSOLE) > specify the path to the expanded Support Pack files, such as SYS:\NW5SP4.

6 Select options.

Follow the online instructions to install the support pack.

During installation, the support pack might prompt you concerning extending the schema. Although you have already extended the schema for eDirectory, you most likely need to extend the schema for other functionality, such as Novell Licensing Services.

7 Take the server down, then restart it.

If you checked the Reboot Server option in Step 6, the server automatically restarts.

Installing eDirectory

1 (Conditional) If you are upgrading eDirectory, do the following:

1a In the AUTOEXEC.NCF file, comment out the lines that load virus scanners, database applications such as Sybase* or Oracle*, backup applications, and other programs that rely on files being continually open and volumes being mounted.

During the eDirectory installation, the software must dismount volumes so that trustee assignments can be migrated.

Be aware that virus scanners and other programs might be embedded inside other products, for example, ZENworks™, ManageWise™, and BorderManager™.

1b Restart the server and verify that the programs and applications referred to in Step 1a are not running.

HINT: If you uncompress the volume you are installing eDirectory on, the install will finish quicker.

2 (Conditional) If you have an IP-only environment, load IPXSPX.NLM.

NWCONFIG.NLM looks to Btrieve* for the product list. Btrieve subsequently requires IPX™. Loading IPXSPX.NLM allows Btrieve to load. When you reboot the server, IPXSPX.NLM does not reload, so you have an IP-only environment again.

3 At the server console, load NWCONFIG.NLM.

4 Select Product Options > Install a Product Not Listed.

5 Press F3 (F4 if you're using RCONSOLE) > enter the path to the eDirectory files under the NW directory, for example, SYS:\NW.

Follow on screen prompts concerning license agreements, the readme file, and tips.

After files are copied, the server automatically restarts and begins to install components for ConsoleOne and Novell Certificate Server. For more information on Novell Certificate Server, see "Understanding the Novell Certificate Server" on page 75.

6 Enter the administrator's login name (for example, Admin.VMP).

IMPORTANT: This window might close before you enter this information. If it does, toggle (Alt+Esc) to the screen and enter the information. Otherwise, the installation will not be complete.

7 Follow the online instructions concerning the Certificate Server, LDAP, languages, components, and products to install.

8 When the installation is completed, restore the lines that you commented out in Step 1a on page 24 > restart the server by clicking Yes.

Repeat this procedure for each NetWare server you want to upgrade to eDirectory 8.6 for NetWare.

Lost Trustee Assignments on NFS Gateway Volumes

The eDirectory installation process does not upgrade trustee assignments on NFS Gateway volumes. If you are hosting NFS Gateway volumes on a server upgraded to eDirectory, those trustee assignments are mapped to non-existent trustees.

To delete the inaccurate trustee assignments, complete the following steps:

1 On the server, load UNICON > authenticate to eDirectory.

2 Select Start/Stop Services > NFS Gateway Server > Del.

3 From a workstation, log in to the server > delete the file SYS:\NFSGW\SFS*xxxx*.DAT.

4 At the server, load UNICON again > authenticate to eDirectory.

5 Select Start/Stop Services > NFS Gateway Server.

You will need to manually create new trustee assignments for eDirectory objects to any NFS Gateway volumes.

Installing eDirectory for Windows NT/2000 Server

Novell eDirectory for NT upgrades NT servers running NT Service Pack 4. If upgrading from previous version of eDirectory, it must be Directory Services version 8.35 or higher.

NOTE: eDirectory runs on Windows 2000 Server, but does not run on Windows 2000 Professional.

If no eDirectory tree exists, you can install eDirectory 8.6. The installation program creates an eDirectory tree.

If your eDirectory tree does not have a Novell Certificate Server, the eDirectory installation program does the following:

- Creates a Security container object for the entire eDirectory tree.

  This object is created at the top of the eDirectory tree and must remain there.

- Creates an Organizational Certificate Authority (CA) object.

- Places the Organizational CA object in the Security container.

Only one Organizational CA object can exist in an eDirectory tree. Because you must not move this object from one server to another, ensure that the first eDirectory server is the one that you intend to permanently host the Organizational CA object. For more information, see "Understanding the Novell Certificate Server" on page 75.

System Requirements

- ❑ A Windows NT server 4.0 with Service Pack 4 or later (or Windows 2000 Server) and an assigned IP address

- ❑ A Pentium 200 with a minimum of 64 MB RAM (128 MB recommended) and a monitor color palette set to a number higher than 16.

- ❑ (Optional) One or more workstations running one of the following:

  - Novell Client for Windows 95/98 3.0 or later

  - Novell Client for Windows NT 4.5 or later

  - NT Client

- ❑ Administrative rights to the NT/2000 server and to all portions of the eDirectory tree that contain domain-enabled User objects. For an installation into an existing tree, you need administrative rights to the Tree object so that you can extend the schema and create objects.

Prerequisites

❑ Because NTFS provides a safer transaction process than a FAT file system provides, you can only install eDirectory on an NTFS partition. Therefore, if you only have FAT file systems, do one of the following:

◆ Create a new partition and format it as NTFS.

Use Disk Administrator. Refer to *Windows NT Server User Guide* for more information.

◆ Convert an existing FAT file system to NTFS, using the CONVERT command.

If your server only has a FAT file system and you forget or overlook this process, the installation program prompts you to provide an NTFS partition.

❑ If you are installing eDirectory for NT into an eDirectory tree that has NetWare and NT servers, each NetWare server must be running one of the following:

◆ NetWare 4.2 with NDS 6.09 or later

◆ NetWare 5.0 with Support Pack 5 or later (http://support.novell.com/misc/patlst.htm#nw)

◆ NetWare 5.1 Support Pack 1 or later.

◆ NetWare 6

Each NT server must be running NDS eDirectory 8.0 or later.

Updating the eDirectory Schema for NT/2000

To upgrade an existing tree, run DSREPAIR on the server that contains the master replica of the Tree partition.

IMPORTANT: If the master replica of the Tree partition resides on a NetWare server, follow the instructions in "Updating the eDirectory Schema for NetWare" on page 21.

The installation program checks for the existing schema's version. If the schema has not been upgraded, the installation program instructs you to run DSREPAIR, then discontinues.

1 Copy PATCHES\DSREPAIR\ NTNDS8\DSREPAIR. DLL from the product CD to the directory, where you installed eDirectory, for example, G:\NOVELL\NDS. This file is version 8.35.

2 Start NDSCONSOLE by running NDSCONS.EXE.

This file is in the directory where you installed eDirectory.

3 Select DSREPAIR from the NDS Service list.

4 Enter `-ins` in the Startup Parameters field > click Start.

After the schema has been updated, the status field next to the DSREPAIR module in NDSCONSOLE will be blank.

5 To see the results of the schema update, select DSREPAIR in NDSCONSOLE.

6 Click Start > File > Open Log File > Open.

The last entry of the log file will contain the results of the schema update.

Installing eDirectory

1 At the NT/2000 server, log in as Administrator or as a user with administrative privileges.

2 Run SETUP.EXE from the NT directory on the product CD.

3 Select which components to install.

You can install the components separately or together.

* Install Novell Directory Services®

Installs eDirectory in an NT-only or mixed NetWare/NT server environment.

Follow the online instructions in the Installation Wizard.

* SLP Directory Agent

Installs SLP Directory Agent, which allows you to control the collection and dissemination of network service information through advanced features.

Follow the online instructions in the Installation Wizard. Select the type of setup you want to install:

Directory: Use eDirectory to manage, configure, and store Directory Agents, scopes, and services.

Local: The Directory Agent and its associated scopes and services are stored and configured through the local machine.

◆ Install ConsoleOne

Installs ConsoleOne 1.3.3.

Follow the online instructions in the Installation Wizard.

The installation program checks for the following components. If a component is missing or is an incorrect version, the installation program automatically launches an installation for the component.

◆ Novell Client for Widows NT/Windows 2000.

For more information on the Novell Client for Windows NT, see the Novell Client for Windows online documentation (http://www.novell.com/documentation/lg/client/docui/index.html).

◆ Novell licensing

You can obtain an evaluation license file from the Novell eDirectory Eval License Download (http://www.novell.com/products/edirectory/licenses/eval_861.html) Web site.

Installing Novell eDirectory on Linux or Solaris

This section will help you install and get started using Novell eDirectory on a Linux or Solaris system. For more information on the Linux or Solaris packages for Novell eDirectory, see "Linux and Solaris Packages for Novell eDirectory" on page 45.

The following installation instructions will help you install and get started using Novell eDirectory on Linux or Solaris:

◆ "System Requirements" on page 29
◆ "Prerequisites" on page 30
◆ "Installing eDirectory" on page 31

System Requirements

Linux

❑ Linux 2.2 and glibc 2.1.3 (glibc 2.2 recommended)
❑ A minimum of 64 MB RAM (128 MB recommended)
❑ 74 MB of disk space for every 50,000 users

❑ ConsoleOne requirements:

 ◆ ConsoleOne 1.3.3

 ◆ A minimum of 64 MB RAM (128 MB recommended)

 ◆ 200 MHz processor (a faster one is recommended)

 ◆ 32 MB disk space

Solaris

❑ All recommended Solaris OS patches are available at the SunSolve*
Online Web page (http://sunsolve.sun.com).

 ◆ Solaris 2.6 (with patch 105591-07 or later)

 ◆ Solaris 7 (with patch 106327-06 or later for 32-bit systems)

 ◆ Solaris 7 (with patch 106300-07 or later for 64-bit systems)

 ◆ Solaris 8

❑ A minimum of 64 MB RAM (128 MB recommended)

❑ 74 MB of disk space for every 50,000 users

❑ ConsoleOne requirements:

 ◆ ConsoleOne 1.3.3

 ◆ A minimum of 64 MB RAM (128 MB recommended)

 ◆ 32 MB disk space

Prerequisites

eDirectory must be installed on all servers that you want to place an
eDirectory replica on.

❑ Meet the platform specific "System Requirements" on page 29.

❑ Enable the Linux or Solaris host for multicast routing. To check if enabled
for multicast routing:

 ◆ On Linux systems, enter the following command:

   /bin/netstat -nr

   The following entry should be present in the routing table:

   224.0.0.0 0.0.0.0

If the entry is not present, log in as root, and enter the following command to enable multicast routing:

route add -net 224.0.0.0 netmask 240.0.0.0 dev -interface

◆ On Solaris systems, enter the following command:

```
/usr/bin/netstat -nr
```

The following entry should be present in the routing table:

```
224.0.0.0 host_IP_address
```

If the entry is not present, log in as root, and enter the following command to enable multicast routing:

```
route add -net 224.0.0.0 -net 224.0.0.0 netmask
    240.0.0.0 hme0
```

❑ NICI 2.3 is a prerequisite for eDirectory installation. eDirectory prompts for the installation of NICI 2.3 during installation if it is not already installed. The package containing NICI 2.3 is named NOVLniu0-2.3.0.

❑ For secure Novell eDirectory operations, you will need the NICI Foundation Key file. You can obtain an evaluation file from the Novell eDirectory Eval License Download (http://www.novell.com/products/ edirectory/licenses/eval_861.html) Web site. If you do not use the NICI Foundation Key, you will not be able to create Certificate Authority and Key Material objects.

❑ If you have more than one server in the tree, the time on all the network servers should be synchronized. Use Network Time Protocol (NTP) to synchronize time. If you want to synchronize time on Linux or Solaris systems with NetWare servers, use TIMESYNC.NLM 5.09 or later. For more information, see "Synchronizing Network Time" on page 80.

❑ If you are installing a secondary server, all the replicas in the partition that you install the product on should be in the On state.

Installing eDirectory

The following sections provide information about installing and uninstalling Novell eDirectory on Linux or Solaris systems:

SLP Usage with eDirectory

If you plan to use SLP to resolve tree names, it should have been properly configured and SLP DAs should be stable. If you don't want to (or cannot) use SLP, you can use a flat file to resolve tree names to server referrals. Refer to man page host.nds for details.

Using the nds-install Utility to Install eDirectory Components

Use the nds-install utility to install eDirectory components on Linux and Solaris systems. This utility is located in the Setup directory on the CD for the respective platforms. The utility adds the required packages based on what components you choose to install.

To install eDirectory components:

1 Log in as root on the host.

2 Enter the following command:

```
nds-install
```

3 When prompted, accept the license agreement.

The installation program displays a list of eDirectory components that you can install.

4 Specify the option for the component you want to install.

Based on the component you choose to install, the installation program proceeds to add the appropriate RPMs, or packages into the Linux or Solaris system. Table 5 lists the packages installed for each eDirectory component.

Table 5   eDirectory Component Packages

eDirectory Component	Packages Installed	Description
eDirectory Server	"NDSbase" on page 45 "NDScommon" on page 46 "NDSsecur" on page 46 "NDSsecutl" on page 46 "NDSmasv" on page 46 "NDSserv" on page 46 "NDSimon" on page 47 "NDSrepair" on page 47 "NDSslp" on page 47 "NDSdexvnt" on page 47	The eDirectory replica server will be installed on the specified server.
Administration Utilities	"NDSadmutl" on page 45 "NDSbase" on page 45 "NDSsecutl" on page 46 "NLDAPbase" on page 47	The Novell Import Conversion Export and LDAP Tools administration utilities will be installed on the specified workstation.
Management Console for eDirectory	"NDSbase" on page 45 "NDSslp" on page 47 "NOVLC1" on page 47 "C1JRE" on page 47 "NDS set of packages" on page 47	The management console for eDirectory will be installed on the specified workstation.
NMAS (Novell Modular Authentication Service) server	"NDSnmas" on page 47	The NMAS server components and the configuration utilities will be installed on the specified server.
NSAdexvnt	"NDSdexvnt" on page 47	This package contains the library that manages events generated in Novell eDirectory to other database.

5 If you are prompted, enter the complete path to the NICI Foundation Key file.

You will be prompted to enter the complete path to the NICI Foundation Key only if the installation program cannot locate the file in the default location (/var, the mounted license diskette, or the current directory).

If the path you entered is not valid, you will be prompted to enter the correct path.

You can use the ndsconfig utility to configure eDirectory Server after installation. However, to do so, you need to ensure that the .nfk file has been copied to the /var directory.

If you have selected Novell Modular Authentication Service™ (NMAS™) as one of the components to be installed, you can use the nmasconfig utility to configure NMAS server after installation. This must be done after configuring eDirectory with ndsconfig.

For more information on the ndsconfig utility, see "ndsconfig Utility" on page 48.

For more information on the nmasconfig utility, see "Using the nmasconfig Utility to Configure NMAS Server" on page 38.

IMPORTANT: Before you begin to use eDirectory, you must ensure that SLP has been installed properly in order for the eDirectory tree to be advertised correctly. To determine if the eDirectory tree is advertised, type the following:

```
/usr/bin/slpinfo -s "ndap.novell//(svcname-
  ws==*tree_name.)/"
```

To install eDirectory components, use the following syntax:

```
nds-install -c component1 [[-c component2]...] [-h] [-n
path_to_.nfk]
```

If you do not provide the required parameters in the command line, the nds-install utility will prompt for the parameters.

Table 6 provides a description of the parameters of the nds-install utility:

Table 6   **The nds-install Utility Parameters**

nds-install Parameter	Description
-c	Specifies the component to be installed based on the packages available. You can install more than one component by using the -c option multiple times.
-h	Specifies the option to display help.

nds-install Parameter	Description
-n	Specifies the path to the file that contains the Novell Foundation Key (.nfk).

Example

To install eDirectory Server packages, enter the following command:

```
nds-install -c server -n /var
```

Using the ndsconfig Utility to Add or Remove the eDirectory Replica Server

You must have Administrator rights to use the ndsconfig utility. When this utility is used with arguments, it validates all arguments and prompts for the password of the user having administrator rights. If the utility is used without arguments, ndsconfig displays a description of the utility and available options. This utility can also be used to remove the eDirectory Replica Server and change the current configuration of eDirectory Server. For more information, see .

To create a new tree:

1 Use the following syntax:

```
ndsconfig new [-m <modulename>] [-i] [-s <servername>] [-
t <tree_name>] [-n <context>] [-d path_for_DIB] [-L
<ldap_port>] [-l <ssl_port>] [-c <caname>[:<casize>]] [-k
<kmoname>[:<kmosize>]] [-e] -a admin_name
```

A new tree is installed with the specified tree name and context. If the parameters are not specified in the command line, ndsconfig prompts you to enter values for each of the missing parameters.

Or, you can also use the following syntax:

```
ndsconfig def [-m <modulename>] [-i] [-s <servername>] [-
t <tree_name>] [-n <context>] [-d path_for_DIB] [-L
<ldap_port>] [-l <ssl_port>] [-c <caname>[:<casize>]] [-k
<kmoname>[:<kmosize>]] [-e] -a admin_name
```

A new tree is installed with the specified tree name and context. If the parameters are not specified in the command line, ndsconfig takes the default value for each of the missing parameters.

To add a server into an existing tree:

1 Use the following syntax:

```
ndsconfig add [-m <modulename>] [-s <servername>] [-p
IP_address] [-t tree_name] [-n context] [-d path_for_DIB]
[-L <ldap_port>] [-l <ssl_port>] [-e] -a admin_name
```

A server is added to an existing tree in the specified context. If the context to which the user wants to add the Server object does not exist, ndsconfig creates the context and adds the server.

LDAP and security services can also be added after eDirectory has been installed into the existing tree.

To remove a Server object and directory services from a tree:

1 Use the following syntax:

```
ndsconfig rm -a admin_name
```

eDirectory and its database are removed from the server.

Table 7   The ndsconfig Utility Parameters

ndsconfig Parameter	Description
new	Creates a new eDirectory tree. If the parameters are not specified in the command line, ndsconfig prompts you to enter values for each of the missing parameters.
def	Creates a new eDirectory tree. If the parameters are not specified in the command line, ndsconfig takes the default value for each of the missing parameters.
add	Adds a server into an existing tree.
rm	Removes the Server object and directory services from a tree.
-i	Ignores a tree of the same name, while installing a new tree. This option is generally not recommended for use.
-s	Specifies the server name.
-t	The tree name to which the server has to be added. If not specified, ndsconfig uses the tree name from the n4u.base.tree-name parameter specified in the etc/nds.conf file. For more information, see "n4u.base.tree-name" on page 50.

ndsconfig Parameter	Description
-n	The context of the server into which the Server object is added. If not specified, ndsconfig uses the context from the n4u.nds.server-context parameter specified in the /etc/nds.conf file. For more information, see "n4u.nds.server-context" on page 51.
-d	The directory path where the database files will be stored.
-L	The TCP port number on the LDAP server.
-l	The SSL port number on the LDAP server.
-c	The name and size of the organizational CA.
-k	The name and size of the Key Material Object.
-a	Distinguished name of the User object that has Supervisor rights to the context in which the Server object and directory services will be created.
-e	Enables clear text password for LDAP objects.
-p	Installs eDirectory Server into an existing tree by specifying the IP address of a server hosting the tree. If this option is used, SLP is not used for tree lookup.
-m	Specifies the module name to install. While installing a new tree, you can install only the ds module. After installing the ds module, you can add the LDAP and SAS services using the add command. If the module name is not specified, all three modules are installed.
set	Sets the value for the specified eDirectory configurable parameters.
get	Lets you view the current value of the eDirectory configurable parameters.
get help	Lets you view the help strings for the eDirectory configurable parameters.

Examples

To create a new tree, enter the following command:

```
ndsconfig new -t corp-tree -n o=company -a
  cn=admin.o=company
```

To add a server into an existing tree, enter the following command:

```
ndsconfig add -t corp-tree -n o=company -a
  cn=admin.o=company
```

To remove the eDirectory Server object and directory services from a tree, enter the following command:

```
ndsconfig rm -a cn=admin.o=company
```

Using the nmasconfig Utility to Configure NMAS Server

This section discusses the following topics:

To use the nmasconfig utility for server configuration, ensure that you have administrative rights to the Security container.

About the nmasconfig Utility

Use this utility for login sequence management, login method management, and Simple Password management, apart from configuring and unconfiguring the NMAS server.

When this utility is used with arguments, it validates them and prompts for the password of the user who has administrative rights.

If the utility is used without arguments, nmasconfig displays a description of the utility and available options.

NOTE: Do not install the current release of NMAS on UNIX on the servers that are part of the tree and have NMAS Enterprise Edition installed and configured on them.

Four modes of operation are available:

1. config

2. method

3. sequence

4. passwd

Only one of the modes of operation can be selected.

For more information on the parameters of nmasconfig utility, see the nmasconfig manpage, *nmasconfig.1m*.

Table 8   **The nmasconfig Utility General Parameters**

nmasconfig Parameter	Description
-t	Refers to the name of the eDirectory tree on which NMAS has to be configured.
	This is an optional parameter. By default, this is taken from the tree name of the current server, read from nds.conf file.
-h *host name*:[*port*]	Refers to the hostname and, optionally, the eDirectory port. By default, this is taken from the hostname of the current server and the default eDirectory port.

Configuring the NMAS Server

The nmasconfig utility lets you configure the NMAS server or remove the configuration of the NMAS server. To configure the server, you must have Administrator rights to the Security container.

Table 9   **The nmasconfig Utility Configure Parameters**

Configure Parameter	Description
-c	Configures NMAS.
-d	Removes NMAS configuration.

Configure Parameter	Description
-a	Refers to the fully distinguished name of the eDirectory administrator with supervisor rights to the security container.
	The fully distinguished name of the administrator should be specified in the typeless, dot-delimited, form without the tree name. This parameter is required.

To Configure the NMAS Server

To configure the NMAS server, enter the following command:

nmasconfig config [**-t** *treename*] [**-h** *hostname*[:*port*]] **-c -a** *adminname*

Example:

To configure NMAS in the tree ACME running on the same host, enter the following command.

nmasconfig config -t *acme* **-c -a** *admin.company*

NOTE: When configuring a remote server, it is recommended to enter the tree and the server name as command line parameters to nmasconfig.

To Remove the NMAS Configuration

To remove the NMAS server configuration, enter the following command:

nmasconfig config [**-t** *treename*] [**-h** *hostname*[:*port*]] **-d -a** *adminname*

Example:

To remove the configuration of NMAS in the tree ACME, enter the following command:

nmasconfig config -t *acme* -d -a *admin.company*

NOTE: Removing the NMAS configuration using the **-d** option does not remove any objects in the Security container. It specifically removes the server-specific configuration.

For NMAS configuration or unconfiguration to take effect, restart the Novell eDirectory server.

Login Method Management

Use the method mode of the nmasconfig utility to install a new login method or upgrade an existing login method to the tree. It can also be used to remove an existing login method.

IMPORTANT: The method binaries for all the platforms can be found in the NMAS directory *nmasmethods*, located on the eDirectory CD. The methods for all platforms can be installed together from ConsoleOne (Windows) or nmasconfig (UNIX).

Installing a New Login Method

To install a new login method or upgrade an existing login method to the tree, enter the following command:

nmasconfig method [general options] **-i | -U -f** *path-to-config.txt* **-a** *admin_name*

Table 10   **The nmasconfig Utility Method Parameters**

Method Parameter	Description
-i	Installs a new method.
	This also creates a login sequence which contains only this login method.
-u	Upgrades NMAS configuration.
-r	Removes an existing method from the tree.
	This also removes the sequence with only this login method, created during this method install.
-a	Refers to the fully distinguished name of the eDirectory administrator with supervisor rights for the context in which the server object and Directory services are to be created.
	The fully distinguished name of the administrator should be specified in the typeless, dot-delimited, form without the tree name. This parameter is required.

Method Parameter	Description
-f	Refers to the absolute or relative path, including the filename, to the config.txt file for the method that needs to be installed. This text file is located in the NMAS methods directory on the install CD. This is a required parameter if either the -i or -U options are specified.
-m	Refers to the name of the NMAS method object that needs to be removed from the tree. If there are spaces or special characters in the method object name, the name should be entered in quotes (" "). This is a required parameter if the -r option is specified.

Example:

To install a new method on the tree running on the current server, enter the following command:

nmasconfig method -i -f *./SimplePassword/config.txt* **-a** *admin.company*

Removing an Existing Login Method

To remove an existing login method, enter the following command:

nmasconfig method [general options] **-r -m** *methodname* **-a** *admin_name*

To remove an existing login method from the tree running on the current server, enter the following command:

nmasconfig method -t *ACME* **-r -m** "*X.509 Certificate*" **-a** *admin.company*

IMPORTANT: You only need to specify one of these options: -i, -U, or -r.

Login Sequence Management

Use the sequence mode of the nmasconfig utility to manage the login sequence.

To manage the login sequence, enter the following command:

nmasconfig sequence [general options] **-D** *user_name* **-a**
admin_name

Table 11   The nmasconfig Utility Sequence Parameters

Sequence Parameter	Description
-D	Refers to the distinguished name of the user object for which sequence management is to be done.
	The user's distinguished name should be specified in the typeless, dot-delimited form without the tree name. This is a required parameter.
-a	Refers to the distinguished name of the user object with supervisor rights to the context in which the previously specified user object is to be modified.
	The admin DN should be specified in the typeless, dot-delimited form without the tree name. This is a required parameter.

Example: To manage the authorized and default sequences of the user named user1 in tree ACME, enter the following command:

nmasconfig sequence -t *ACME* **-D** *user1.finance.company* **-a**
admin.company

The sequence management option generates a menu of options.

Table 12   The Sequence Management Menu Options

Sequence Management Options	Description
(a) Authorize a method	Authorizes a sequence present in the Available Login Sequences list.
(b) Remove an authorized method	Removes an existing authorized login sequence from the Authorized Login Sequences list.
(c) Change default login sequence	Sets the default login sequence for the user from the Authorized Login Sequences list.
(d) Commit current changes and exit	Commits the changes to eDirectory and exits the sequence management menu.

Sequence Management Options	Description
(e) Quit without saving	Quits the sequence management menu without saving the changes.

IMPORTANT: The nmasconfig utility does not have the functionality to create, delete, or modify sequences. The sequence mode supports only the operations which are performed for a user, such as setting the Authorized or default sequences. For other sequence operations, use ConsoleOne on a Windows workstation.

Managing Simple Passwords

Use the passwd mode of the nmasconfig utility to set simple passwords.

To set the simple password for a specified user in the tree, enter the following command:

nmasconfig passwd [general options] [**-H** *hash_type*] [**-a** *admin_name*] **-D** *user_name*

Table 13   **The nmasconfig Utility Password Parameters**

Password Parameter	Description
-H	Refers to the hashing format in which the simple password for the user needs to be stored in eDirectory. The valid values are "sha," "md5," or "clear."
	By default, the simple password hash type is "clear."
-a	Refers to the distinguished name of the user object with supervisor rights to the context where the specified User object's simple password is to be modified.
	The administrator's fully distinguished name should be specified in the typeless, dot-delimited form without the tree name.
-D	Refers to the distinguished name of the user object for which simple password change is to be done. The user DN should be specified in the typeless, dot-delimited form without the tree name. This is a required parameter.

Example 1: If you are an admin and are changing another user's simple password, enter the following command:

nmasconfig passwd -a *admin.company* **-D** *user1.finance.company*

Example 2: If you are modifying your own simple password, enter the following command:

nmasconfig passwd -D *user1.finance.company*

Linux and Solaris Packages for Novell eDirectory

Novell eDirectory includes a Linux or Solaris package system, which is a collection of tools that simplify the installation and uninstallation of various eDirectory components. Packages contain makefiles that describe the requirements to build a certain component of eDirectory. Packages also include configuration files, utilities, libraries, daemons, and manual pages that use the standard Linux or Solaris tools installed with the OS.

Table 14 on page 45 provides information about the Linux and Solaris packages that are included with Novell eDirectory.

Table 14   Linux and Solaris Packages for Novell eDirectory

Package	Description
NDSadmutl	Contains the Novell Import Conversion Export utility and is dependent on the NDSbase package.
NDSbase	Represents the Directory User Agent. This package is dependent on the NDSslp package. The NDSbase package contains the following: ◆ An authentication toolbox containing the RSA authentication needed for eDirectory ◆ A platform-independent system abstraction library, a library containing all the defined Directory User Agent functions, and the schema extension library ◆ A combined configuration utility and the Directory User Agent test utility ◆ The eDirectory configuration file and manual pages

Package	Description
NDScommon	Contains the man pages for the eDirectory configuration file, install, and uninstall utilities.
NDSsecur	Contains the server-side security components eDirectory Server uses, including the following: ◆ SAS SDK, and PKI Server ◆ Java Wrapper over SAS SDK ◆ Pure Java SSL
NDSsecutl	Contains the client-side security utilities, including the following: ◆ PKI Client and configuration utility
NDSmasv	Contains the libraries required for mandatory access control (MASV).
NDSserv	Contains all the binaries and libraries needed by the eDirectory Server. It also contains the utilities to manage the eDirectory Server on the system. This package is dependent on the NDSbase and NDSsecur packages. The NDSserv package contains the following: ◆ An NDS install library, FLAIM library, trace library, NDS library, LDAP server library, LDAP install library, index editor library, DNS library, merge library, and LDAP extension library for LDAP SDK ◆ eDirectory Server daemon ◆ A binary for DNS and a binary to load or unload LDAP ◆ The utility needed to create the MAC address, the utility to trace the server and change some of the global variables of the server, the utility to back up and restore eDirectory, and the utility to merge eDirectory trees ◆ Startup scripts for DNS, NDSD, and NLDAP ◆ Manual pages

Package	Description
NDSimon	Contains the runtime libraries and utilities used to search and retrieve data from eDirectory services. This package is dependent on the NDSbase package.
NDSrepair	Contains the runtime libraries and utility which corrects problems in the eDirectory database. This package is dependent on the NDSbase package.
NDSslp	The NDSslp package contains the following: ◆ The SLP User Agent/Service Agent daemon and the SLP libraries to access SLP ◆ The transport library, utility library, and configuration library that the SLP daemon uses ◆ The Unicode* library that the SLP daemon and API library use.
NLDAPbase	Contains LDAP libraries, extensions to LDAP libraries, and the following LDAP tools: ◆ ldapdelete ◆ ldapmodify ◆ ldapmodrdn ◆ ldapsearch ◆ ndsindex
NDS set of packages	Contains a set of ConsoleOne snap-ins.
NOVLC1	Contains Linux or Solaris package for the ConsoleOne management utility.
C1JRE	Contains the Java* runtime files and libraries that are required to run ConsoleOne on Linux or Solaris systems.
NDSnmas	Contains all the NMAS libraries, and nmasconfig binaries needed for NMAS server.
NDSdexvnt	This package contains library that manages events generated in Novell eDirectory to other database.

Configuring Novell eDirectory on Linux or Solaris Systems

Novell eDirectory includes configuration utilities that simplify the configuration of various eDirectory components. The following sections provide information about functionality and usage of Novell eDirectory configuration components:

Configuration Utilities

This section provides information about using the following eDirectory configuration utilities:

ndsconfig Utility

You can use the ndsconfig utility to configure eDirectory. This utility can also be used to add the NDS Replica Server into an existing tree or to create a new tree. It can also be used to remove the NDS Replica Server. For more information, see "Using the ndsconfig Utility to Add or Remove the eDirectory Replica Server" on page 35.

To change the current configuration of the installed components:

1 Use the following syntax:

```
ndsconfig {set value_list | get [parameter_list] | get
help [parameter_list]}
```

Refer to Table 7 on page 36 for a description of ndsconfig parameters.

ldapconfig Utility to Configure the LDAP Server and LDAP Group Objects

You can use the LDAP configuration utility, ldapconfig, on Linux and Solaris systems to modify, view, and refresh the attributes of LDAP Server and Group objects. For more information, see "Configuring LDAP Server and LDAP Group Objects on Linux or Solaris Systems" on page 339.

nmasconfig Utility To Configure Novell Modular Authentication Service

You can use nmasconfig to configure and unconfigure an NMAS server on Solaris and Linux. For more information about configuring and unconfiguring NMAS, see "Using the nmasconfig Utility to Configure NMAS Server" on page 38.

The nmasconfig utility can also be used to manage login methods and login sequence, and to change simple passwords. For more information on the parameters of the nmasconfig utility, see the nmasconfig manpage, *nmasconfig.1m*.

pkiconfig Utility to Create Novell PKI Objects for eDirectory

The pkiconfig utility lets you create the following PKI objects for eDirectory:

- Organizational Certificate Authority (CA) for an eDirectory tree
- Server Certificates (Stored as Key Material Objects (KMO))
- User Certificates
- Security Container for an NDS tree
- SAS Service object

For information on using pkiconfig utility, see the pkiconfig manpage, *pkiconfig.1m.*

Configuration Parameters

The eDirectory configuration parameters are stored in the nds.conf file.

Table 15 on page 50 provides a description of these parameters.

Table 15　　**eDirectory Configuration Parameters**

eDirectory Configuration Parameter	Description
n4u.base.tree-name	The tree name that Account Management uses. This is a mandatory parameter set by the Account Management installer. This parameter cannot be set or changed by the administrator.
n4u.base.dclient.use-udp	The Directory User Agent can use UDP in addition to TCP for communicating with eDirectory servers. This parameter enables the UDP transport. The default value is 0. The range is 0 or 1.
n4u.base.slp.max-wait	The Service Location Protocol (SLP) API calls time out. The default value is 30. The range is 3 - 100.
n4u.uam.preferred-server	The host name of the machine that hosts the eDirectory service. The Directory User Agent can use a preferred server, if one is available. The preferred server has to be set to any of the servers hosting a master or read/write replica. If the eDirectory replica is present on the Linux or Solaris system, set the preferred server to the hostname of the Linux or Solaris system for efficiency. The default value is null.
n4u.nds.advertise-life-time	eDirectory re-registers itself with the Directory Agent after this time period. The default value is 3600. The range is 1-65535.
n4u.server.signature-level	The Signature Level determines the level of enhanced security support. Increasing this value increases security, but decreases performance. The default value is 1. The range is 0-3.

eDirectory Configuration Parameter	Description
n4u.nds.dibdir	The eDirectory directory information database. The default value is /var/nds/dib. This parameter is set during installation and cannot be modified later.
n4u.nds.server-name	The name of the eDirectory Server. The default value is null.
n4u.nds.bindery-context	The Bindery Context string. The default value is null.
n4u.nds.server-context	The context into which the eDirectory server is added. This parameter cannot be set or changed.
n4u.nds.external-reference-life-span	The number of hours unused external references are allowed to exist before being removed. The default value is 192. The range is 1-384.
n4u.nds.inactivity-synchronization-interval	The interval, in minutes, after which full synchronization of the replicas is performed, following a period of no change to the information held in eDirectory on the server. The default value is 60. The range is 2-1440.
n4u.nds.synchronization-restrictions	The Off value allows synchronization with any version of eDirectory. The On value restricts synchronization to version numbers you specify as parameters, for example, ON,420,421. The default value is Off.
n4u.nds.janitor-interval	The interval in minutes after which the eDirectory janitor process is executed. The default value is 2. The range is 1-10080.

eDirectory Configuration Parameter	Description
n4u.nds.backlink-interval	The interval, in minutes, after which eDirectory backlink consistency is checked. The default value is 780. The range is 2-10080.
n4u.nds.flatcleaning-interval	The interval, in minutes, after which the flatcleaner process automatically begins purging and deleting entries from the database. The default value is 720. The range is 1-720.
n4u.nds.server-state-up-threshold	The Server State Up threshold, in minutes.This is the time after which eDirectory checks the server state before returning -625 errors. The default value is 30. The range is 1-720.
n4u.nds.heartbeat-schema	The heartbeat base schema synchronization interval in minutes. The default value is 240. The range is 2-1440.
n4u.nds.heartbeat-data	The heartbeat synchronization interval in minutes. The default value is 60. The range is 2-1440.
n4u.nds.drl-interval	The interval, in minutes, after which eDirectory distributed reference link consistency is checked. The default value is 780. The range is 2-10080.
n4u.server.tcp-port	The default port used if the port number is not specified in the n4u.server.interfaces parameter.

eDirectory Configuration Parameter	Description
n4u.server.interfaces	The IP address and port number the eDirectory server should listen to for client connections. The value can be a comma-separated list specifying more than one combination of possible settings. You can specify the value as <Interface name>|<IP Address>@<port>. You can specify either the whole string or the interface name or IP address. If the parameter is not specified in the nds.conf file, the eDirectory server gets any one IP address with the default port specified in the n4u.server.interfaces parameter. If the n4u.server.interfaces parameter is also not specified, the default port will be 524. The possible values are given below.
	◆ *Interface name | IP Address* - The eDirectory server gets the specified interface name or IP address with the default port.
	◆ *Interface name | IP Address@port* - The eDirectory server gets the specified interface name or IP address with the specified port number.
n4u.server.active-interval	A worker thread in the thread pool is active if it is available to execute jobs in the ready queue. This parameter sets the time interval (in milliseconds) within which a thread should return to the thread pool to be considered active. This interval will be scaled internally based on the number of processors configured. The default value is 10,000 milliseconds (10 seconds).

eDirectory Configuration Parameter	Description
n4u.ldap.lburp.transize	The number of records that will be sent from the Novell Import Conversion Export client to the LDAP server in a single LBURP packet. You can increase the transaction size to ensure that multiple add operations can be performed in a single request. The default transaction size is 25. You can provide a transaction size in the hard-limit range of 1 and 250.

Uninstalling eDirectory on NetWare

If necessary, you can remove eDirectory from a NetWare server.

IMPORTANT: Removing eDirectory from a NetWare server makes the NetWare volumes and file system inaccessible.

1 At the server console, enter **load nwconfig**.

2 Select Directory Options > Remove Directory Services from This Server.

3 Follow the online instructions.

Uninstalling eDirectory on Windows NT/2000

You can use the Control Panel to uninstall eDirectory on NT/2000.

1 From the Windows NT/2000 server where eDirectory is installed, click Start > Settings > Control Panel > Add/Remove Programs.

2 Select eDirectory from the list > click Add/Remove.

3 Confirm that you want to remove eDirectory by clicking Yes.

The Installation Wizard removes eDirectory from the server.

To uninstall ConsoleOne or the SLP Directory Agent, repeat the steps above, except in Step 2 select either ConsoleOne or Novell SLP Directory Agent.

Uninstalling eDirectory from Linux or Solaris Systems

Use the nds-uninstall utility to uninstall eDirectory components from Linux or Solaris systems. The uninstall utility uninstalls eDirectory from the local host.

1 Execute the **nds-uninstall** command.

The utility lists the installed components.

2 Select the required component.

The following section provides information about uninstalling eDirectory components:

  ◆ "Using the nds-uninstall Utility to Uninstall eDirectory Components" on page 55

IMPORTANT: If the Linux or Solaris system you are installing Novell eDirectory 8.6 on already has NDS 8.0 components installed on it, you can use the nds80-uninstall utility to uninstall NDS 8.0 components, and the nds-uninstall utility to uninstall eDirectory 8.6 components. If your system already has NDS eDirectory 8.5 components installed on it, you can use the nds85-uninstall utility to uninstall NDS eDirectory 8.5 components. Uninstallation does not remove NICI 2.3.

Using the nds-uninstall Utility to Uninstall eDirectory Components

To uninstall eDirectory components:

1 Use the following syntax:

```
nds-uninstall -c component1 [[-c component2]...] [-h]
```

If you do not provide the required parameters in the command line, the nds-install utility will prompt for the parameters.

Table 16   The nds-uninstall Utility Parameters

nds-uninstall Parameter	Description
-h	Displays the help strings.
-c	Specifies the component that is to be uninstalled. More than one component can be uninstalled by using the -c option multiple times.

Examples

To uninstall eDirectory Server packages, enter the following command:

```
nds-uninstall -c server
```

2 Designing Your Novell eDirectory Network

The design of Novell® eDirectory™ impacts virtually every network user and resource. A good eDirectory design can enhance the performance and value of the entire network by making the network more efficient, fault tolerant, secure, and scalable and operable. This section provides suggestions for designing your eDirectory network.

eDirectory Design Basics

An efficient eDirectory design is based on the network layout, organizational structure of the company, and proper preparation.

If you are designing eDirectory for e-business, refer to .

Network Layout

The network layout is the physical setup of your network. To develop an efficient eDirectory design, you need to be aware of the following:

- WAN links
- Users that need remote access
- Network resources, such as number of servers
- Network conditions, such as frequent power outages
- Anticipated changes to the network layout

Organizational Structure

The organizational structure of the company will influence the eDirectory design. To develop an efficient eDirectory design you need:

- The organizational chart and an understanding of how the company operates
- Personnel who have the skills needed to complete the design and implementation of your eDirectory tree

  You will need to identify personnel who can do the following:

    - Maintain the focus and schedule of the eDirectory design
    - Understand eDirectory design, design standards, and security
    - Understand and maintain the physical network structure
    - Manage the internetwork backbone, telecommunications, WAN design, and router placement

Preparing for eDirectory Design

Before you actually create the eDirectory design, you should:

- Set realistic expectations concerning scope and schedule
- Notify all users who will be affected by the design of your implementation of eDirectory
- Obtain the information identified in "Network Layout" on page 57 and "Organizational Structure" on page 58

Designing the eDirectory Tree

Designing the eDirectory tree is the most important procedure in the design and implementation of a network. The design consists of the following tasks:

- "Creating a Naming Standards Document" on page 59
- "Designing the Upper Layers of the Tree" on page 62
- "Designing the Lower Layers of the Tree" on page 65

Creating a Naming Standards Document

Using standard names such as object names makes your network more intuitive to both users and administrators. Written standards can also specify how administrators set other property values, such as telephone numbers and addresses.

Searching and browsing the directory rely greatly on the consistency of naming or property values.

The use of standard names also makes it easier for DirXML™ to move data between eDirectory and other applications. For more information on DirXML, see DirXML Administration Guide (http://www.novell.com/documentation/lg/ndsdirxml/docui/index.html).

Naming Conventions

Objects

- The name must be unique in the container. For example, Debra Jones and Daniel Jones cannot both be named DJONES if they are in the same container.

- Special characters are allowed. However, plus signs (+), equals signs (=), and periods (.) must be preceded by a backslash (\) if used. Additional naming conventions apply to Server and Country objects, as well as to bindery services and multilingual environments.

- Uppercase and lowercase letters, as well as underscores and spaces, are displayed as you first entered them, but they aren't distinguished. For example, Manager_Profile and MANAGER PROFILE are considered identical.

- If you use spaces, you must enclose the name in quotes when entering it on the command line or in login scripts.

Server Objects

- Server objects are automatically created when you install new servers.

- You can create additional Server objects for existing NetWare® and NT servers and for eDirectory servers in other trees, but they are all treated as bindery objects.

- When creating a Server object, the name must match the physical server name, which:

  - Is unique in the entire network

- Is from 2 to 47 characters long

- Contains only letters A-Z, numbers 0-9, hyphens, periods, and underscores

- Doesn't use a period as the first character

- Once named, the Server object cannot be renamed in ConsoleOne™. If you rename it at the server, the new name automatically appears in ConsoleOne.

Country Objects

Country objects should follow the standard two-letter ISO country code.

Bindery Objects

If the object is accessed from NetWare 2 or NetWare 3 through bindery services, the following restrictions apply:

- Spaces in the name are replaced with underscores

- Names are truncated to 47 characters

- The following characters are not allowed: slash (/), backslash (\), colon (:), comma (,), asterisk (*), and question mark (?)

IMPORTANT: Bindery emulation is not supported on Linux* or Solaris* platforms.

Multilingual Considerations

If you have workstations running in different languages, you might want to limit object names to characters that are viewable on all the workstations. For example, a name entered in Japanese cannot contain characters that aren't viewable in Western languages.

IMPORTANT: The Tree name should always be specified in English.

Sample Standards Document

The following is a sample document containing standards for some of the most frequently used properties. You only need to have standards for those properties you use. Distribute the standards document to all administrators responsible for creating or modifying objects.

Table 17   Sample Standards Document

Object Class	Property	Standard	Examples	Rationale
User | Login name	First initial, middle initial (if applicable), and last name (all lowercase). Eight characters maximum. All common names are unique in the company.	msmith, bgashler	Using unique names company-wide is not required by eDirectory but helps avoid conflicts within the same context (or bindery context).
User | Last name	Last name (normal capitalization).	Gashler	Used for generating mailing labels.
Telephone and fax numbers	Numbers separated by dashes.	US: 123-456-7890 Other: 44-344-123456	Used by autodialing software.
Multiple classes | Location	Two-letter location code (uppercase), dash, mail stop.	BA-C23	Used by interoffice mail carriers.
Organization | Name	YourCo for all trees.	YourCo	If you have separate trees, a standard Organization name allows for future merging of trees.
Organizational Unit | Name (based on location)	Two or three-letter location code, all uppercase.	ATL, CHI, CUP, LA, BAT, BOS, DAL	Short, standard names are used for efficient searching.
Organizational Unit | Name (based on department)	Department name or abbreviation.	Sales, Eng	Short, standard names make it easy to identify which department the container is servicing.
Group | Name	Descriptive name.	Project Managers	Avoid extremely long names; some utilities will not display them.
Directory Map | Name	Contents of the directory indicated by the Directory Map.	DOSAPPS	Short, standard names make it easy to identify which department the container is servicing.

Object Class | Property	Standard	Examples	Rationale
Profile | Name	Purpose of the profile.	MobileUser	Short, standard names make it easy to identify which department the container is servicing.
Server | Name	SERV, dash, department, dash, unique number.	SERV-Eng-1	eDirectory requires Server names to be unique in the tree.

Designing the Upper Layers of the Tree

You should carefully design the upper layers of the tree because changes to the upper layers affect the rest of the tree, especially if your organization has WAN links. You want to design the top of the tree so that few changes will be necessary.

Use the following eDirectory design rules to create your eDirectory tree:

- ◆ Use a pyramid design.

- ◆ Use one eDirectory tree with a unique name.

- ◆ Create a single Organization object.

- ◆ Create first-level Organizational Units that represent the physical network infrastructure.

depicts the eDirectory design rules.

Figure 1 eDirectory Design Rules
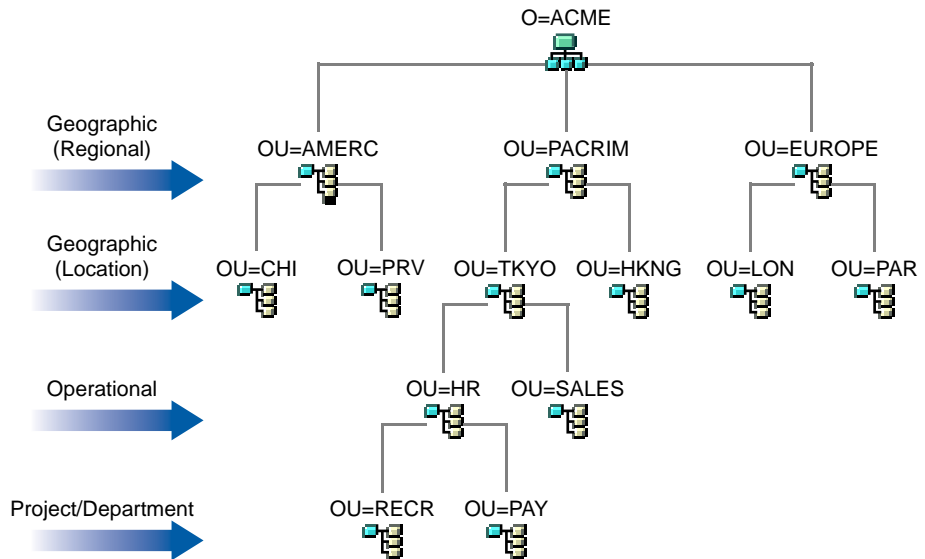
To create the upper layers of the tree, refer to "Creating and Manipulating Objects" in *ConsoleOne User Guide*.

Using a Pyramid Design

With a pyramid-designed eDirectory, managing, initiating changes to large groups, and creating logical partitions are easier.

The alternative to the pyramid design is a flat tree that places all objects in the top layers of the tree. eDirectory can support a flat tree design; however, a flat tree design can be more difficult to manage and partition.

Using One eDirectory Tree with a Unique Name

A single tree works best for most organizations. By default, one tree is created. With one tree you have a single user identity on the network, simpler administration of security, and single point of management.

This recommendation for a single tree for business use does not preclude additional trees for testing and development.

Some organizations, however, might need multiple trees because of legal, political, or corporate issues. For example, an organization consisting of

Designing Your Novell eDirectory Network   **63**

several autonomous organizations might need to create several trees. If your organization needs multiple trees, consider using DirXML to simplify management. For more information on DirXML, see DirXML Administration Guide (http://www.novell.com/documentation/lg/ndsdirxml/docui/index.html).

When you name the tree, use a unique name that will not conflict with other tree names. Use a name that is short and descriptive, such as EDL-TREE.

If two trees have the same name and are located on the same network, you might encounter the following problems:

- ◆ Updates going to the wrong tree
- ◆ Resources disappearing
- ◆ Rights disappearing
- ◆ Corruption

You can change the tree name using the DSMERGE utility, but do so with caution. A tree name change impacts the network because you need to reconfigure the clients to use the new tree name.

Creating a Single Organization Object

Generally, an eDirectory tree should have one Organization object. By default, a single Organization object is created and named after your company. This allows you to configure changes that apply to the whole company from a single location in the tree.

For example, you can use ZENworks™ to create a Workstation Import Policy object in the Organization object. In this policy, which affects the whole organization, you define how Workstation objects are named when created in eDirectory.

In the Organization container, the following objects are created:

- ◆ Admin
- ◆ Server
- ◆ Volume

  Networks with only a Windows* NT* server running eDirectory have no Volume objects.

You might want to create multiple Organization objects if your company has the following needs:

- It comprises multiple companies that do not share the same network.

- It needs to represent separate business units or organizations.

- It has a policy or other internal guidelines that dictate that organizations remain separate.

Creating Organizational Units That Represent the Physical Network

First-level Organizational Unit design is important because it affects the partitioning and efficiency of eDirectory.

For networks that span more than one building or location using either a LAN or a WAN, the first-level Organizational Unit object design should be based on location. This allows you to partition eDirectory in a way that keeps all objects in a partition at one location. It also provides a natural place to make security and administrator assignments for each location.

Designing the Lower Layers of the Tree

You should design the lower layers of the tree based on the organization of network resources. You have more freedom in designing the lower layers of an eDirectory tree than the upper layers because lower-layer design only affects objects at the same location.

To create the lower layers of the tree, refer to "Creating and Manipulating Objects" in *ConsoleOne User Guide*.

Determining Container, Tree, and Database Size

The number of lower-level container objects you create depends on the total number of objects in your tree and your disk space and disk I/O speed limitations. eDirectory has been tested with over 1 billion objects in a single eDirectory tree, so the only real limitations are disk space, disk I/O speed, and RAM to maintain performance. Keep in mind the impact of replication on a large tree.

A typical object in eDirectory is 3 to 5 KB in size. Using this object size, you can quickly calculate disk space requirements for the number of objects you have or need. Keep in mind that the object size will grow depending upon how many attributes are completed with data and what the data is. If objects will hold binary large object (BLOB) data such as pictures, sounds, or biometrics, the object size will subsequently grow.

The larger the partitions, the slower the replication cycles. If you are using products that require the use of eDirectory, such as ZENworks and DNS/DHCP services, the eDirectory objects created by these products will affect the size of the containers they are located in. You might consider placing objects that are for administration purposes only, such as DNS/DHCP, in their own partition so user access is not affected with slower replication. Also, managing partitions and replicas will be easier.

If you are interested, you can easily determine the size of your eDirectory database or the Directory Information Base (DIB) Set.

- For NetWare, download TOOLBOX.NLM from the Novell Support Connection Web site (http://support.novell.com) to see the SYS:_NETWARE directory on your server.

- For Windows, look at the DIB Set at \NOVELL\NDS\DIBFiles.

- For Linux or Solaris, look at the DIB Set in the directory you specified during installation.

Deciding Which Containers to Create

In general, create containers for objects that have access needs in common with other eDirectory objects. This lets you service many users with one trustee assignment or login script. You can create containers specifically to make container login scripts more effective, or you can place two departments in one container to make login script maintenance more feasible.

Keep users close to the resources they need to limit traffic over the network. For example, people who work in the same department generally work near each other. They usually need access to the same file system, and they print to the same printers.

Exceptions to general workgroup boundaries are not hard to manage. If two workgroups use a common printer, for instance, you can create an Alias object to the printer in one of the workgroups. You can create Group objects to manage some User objects within a workgroup or User objects across multiple workgroups. You can create Profile objects for subsets of users with unique login script requirements.

Guidelines for Partitioning Your Tree

When you partition eDirectory, you allow parts of the database to exist on several servers. With this capability, you can optimize network use by distributing the eDirectory data processing and storage load over multiple servers on the network. By default, a single partition is created. For more information on partitions, refer to "Partitions" on page 122. For information on creating partitions, refer to Chapter 6, "Managing Partitions and Replicas," on page 161.

The following are guidelines for most networks. However, depending on the specific configuration, hardware, and traffic throughput of the network, you might need to adjust some guidelines to fit your needs.

Determining Partitions for the Upper Layers of the Tree

Just as you design your tree with a pyramid design, you will also partition with a pyramid design. Your partition structure will have few partitions at the top of the tree and more partitions as you move toward the bottom. Such a design creates fewer subordinate references than an eDirectory tree structure that has more partitions at the top than at the bottom.

This pyramid design can be achieved if you always create the partitions relatively close to the leaf objects, particularly the users. (An exception is the partition created at the root of the tree during installation.)

When designing the partitions for the upper layers, keep the following in mind:

- Partition the top of the tree based on the WAN infrastructure. Place fewer partitions at the top of the tree with more at the bottom.

  You can create containers for each site separated by WAN links (placing each Server object in its local container), then create a partition for each site.

- In a network with WAN links, partitions should not span multiple locations.

  This design ensures that replication traffic between different sites is not unnecessarily consuming WAN bandwidth.

- Partition locally around the servers. Keep physically distant servers in separate partitions.

For more information on managing your WAN traffic, see Chapter 9, "WAN Traffic Manager," on page 285.

Determining Partitions for the Lower Layers of the Tree

When designing the partitions for the lower layers of the eDirectory tree, keep the following in mind:

- ◆ Define lower-layer partitions by organizational divisions, departments, and workgroups, and their associated resources.

- ◆ Partition so that all objects in each partition are at a single location. This ensures that updates to eDirectory can occur on a local server.

Determining Partition Size

With eDirectory, we recommend the following design limits for partition sizes:

Table 18   **Design Limits for Partition Sizes**

Partition Size	Unlimited Objects
	Replica Directory Information Base (DIB) limited to ITB
Total number of partitions in tree	Unlimited
Number of child partitions per parent	150
Number of replicas per partition	50
	Limited by replica DIB
Number of replicas per replica server	250

This change in design guidelines from NDS 6 and 7 is because of architectural changes in NDS 8. These recommendations apply to distributed environments such as corporate enterprises. These recommendations might not subsequently apply to e-business or applications.

Although typical e-business users require that all the data be stored on a single server, eDirectory 8.6 provides filtered replicas that can contain a subset of objects and attributes from different areas of the tree. This allows for the same e-business needs without storing all the data on the server. For more information, see "Filtered Replicas" on page 129.

Considering Network Variables

Consider the following network variables and their limitations when planning your partitions.

- The number and speed of servers
- The speed of network infrastructure, such as network adapters, hubs, and routers
- The amount of network traffic

Guidelines for Replicating Your Tree

Creating multiple eDirectory partitions does not, by itself, increase fault tolerance or improve performance of the directory; however, strategically using multiple replicas does. The placement of replicas is extremely important for accessibility and fault tolerance. eDirectory data needs to be available as quickly as possible and needs to copied in several places to ensure fault tolerance. For information on creating replicas, refer to Chapter 6, "Managing Partitions and Replicas," on page 161.

The following guidelines will help determine your replica placement strategy.

Workgroup Needs

Place replicas of each partition on servers that are physically close to the workgroup that uses the information in that partition. If users on one side of a WAN link often access a replica stored on a server on the other side, place a replica on servers on both sides of the WAN link.

Place replicas in the location of highest access by users, groups, and services. If groups of users in two separate containers need access to the same object within another partition boundary, place the replica on a server that exists in the container one level above the two containers holding the group.

Fault Tolerance

If a disk crashes or a server goes down, replicas on servers in other locations can still authenticate users to the network and provide information on objects in partitions stored on the disabled server.

With the same information distributed on several servers, you are not dependent on any single server to authenticate you to the network or to provide services (such as login).

To create fault tolerance, plan for three replicas for each partition if the directory tree has enough servers to support that number. There should be at least two local replicas of the local partition. There is no need to have more than three replicas unless you need to provide for accessibility of the data at other locations, or you participate in e-business or other applications that need to have multiple instances of the data for load balancing and fault tolerance.

You can have only one master replica. Additional replicas must be read/write, read-only, or filtered. Most replicas should be read/write. They can handle object viewing, object management, and user login, just as the master replica can. They send out information for synchronization when a change is made.

Read-only replicas cannot be written to. They allow object searching and viewing, and they are updated when the replicas of the partition synchronize.

Do not depend on a subordinate reference or filtered replicas for fault tolerance. A subordinate reference is a pointer and does not contain objects other than the partition root object. Filtered replicas do not contain all objects within the partition.

eDirectory allows for an unlimited number of replicas per partition, but the amount of network traffic increases as the number of replicas increase. Balance fault tolerance needs with network performance needs.

You can store only one replica per partition on a server. A single server can store replicas of multiple partitions.

Depending on your organization's disaster recovery plan, the major work of rebuilding the network after a loss of a server or location can be done using partition replicas. If the location has only one server, back up eDirectory regularly. (Some backup software does not back up eDirectory.) Consider purchasing another server for fault tolerance replication.

Determining the Number of Replicas

The limiting factor in creating multiple replicas is the amount of processing time and traffic required to synchronize them. When a change is made to an object, that change is communicated to all replicas in the replica list. The more replicas in a replica list, the more communication is required to synchronize changes. If replicas must synchronize across a WAN link, the time cost of synchronization is greater.

If you plan partitions for many geographical sites, some servers will receive numerous subordinate reference replicas. eDirectory can distribute these subordinate references among more servers if you create regional partitions.

Replicating the Tree Partition

The Tree partition is the most important partition of the eDirectory tree. If the only replica of this partition becomes corrupted, users will experience impaired functionality on the network until the partition is repaired or the eDirectory tree is completely rebuilt. You will also not be able to make any tree design changes involving the Tree.

When creating replicas of the Tree partition, balance the cost of synchronizing subordinate references with the number of replicas of the Tree partition.

Replicating for Administration

Because partition changes only originate at the master replica, place master replicas on servers near the network administrator in a central location. It might seem logical to keep masters at remote sites; however, master replicas should be where the partition operations will occur.

We recommend that major eDirectory operations, such as partitioning, be handled by one person or group in a central location. This methodology limits errors that could have adverse effects to eDirectory operations and provides for a central backup of the master replicas.

The network administrator should perform high-cost activities, such as creating a replica, at times when network traffic is low.

Meeting Bindery Services Needs for NetWare

If you are using eDirectory on NetWare and your users require access to a server through bindery services, that server must contain a master or read/write replica that contains the bindery context. The bindery context is set by the SET BINDERY CONTEXT statement in AUTOEXEC.NCF.

Users can access objects providing bindery services only if real objects exist on that server. Adding a replica of a partition to the server adds real objects to the server and lets users with User objects in that partition log in to the server with a bindery connection.

For more information on bindery services, refer to "NetWare Bindery Emulation" on page 130.

Managing WAN Traffic

If users currently use a WAN link to access particular directory information, you can decrease access time and WAN traffic by placing a replica containing the needed information on a server that users can access locally.

If you are replicating the master replicas to a remote site or are forced to place replicas over the WAN for accessibility or fault tolerance, keep in mind the bandwidth that will be used for replication.

Replicas should only be placed in non-local sites to ensure fault tolerance if you are not able to get the recommended three replicas, increase accessibility, and provide centralized management and storage of master replicas.

To control the replication of eDirectory traffic over WAN links, use WAN Manager. For more information on WAN Manager, refer to Chapter 9, "WAN Traffic Manager," on page 285.

Replica Advisor

If you have Novell Account Management, you can use the Replica Advisor to help you decide how to partition the tree and decide which replicas to place on which servers. You can access the Replica Advisor page from ConsoleOne by viewing the details of a Domain object.

If you are using eDirectory for Windows, the Replica Advisor page of the Domain object shows all the partitions that contain the User objects that have membership in the Domain. When the partition item is expanded, it lists User objects in that partition. For more information, refer to "Using the Replica Advisor" in Novell Account Management Administration Guide (http://www.novell.com/documentation/lg/ndsam/index.html).

Planning the User Environment

After you have designed the basic structure of the eDirectory tree and have set up partitioning and replication, you should plan the user environment to simplify management and increase access to network resources. To create a user environment plan, review the users needs and create accessibility guidelines for each area.

Reviewing Users Needs

When you review users needs, consider the following:

- Physical network needs, such as printers or file storage space

  Evaluate if resources are shared by groups of users within a tree or shared by groups of users from multiple containers. Also consider the physical resource needs of remote users.

- Bindery services needs for NetWare users

  Consider which applications are bindery-based and who uses them.

- Application needs

  Consider which applications and data files are needed by users, what operating systems exist, and which groups or users need access to applications. Consider if the shared applications should be manually or automatically launched by applications such as ZENworks.

Creating Accessibility Guidelines

After you have gathered information about user needs, you should determine the eDirectory objects that you will use to create the users' environments. For example, if you create policy packages or Application objects, you should determine how many you will create and where you will allow them to be placed in the tree.

You should also determine how you will implement security to restrict user access. You should identify any security precautions related to specific security practices. For example, you could warn network administrators to avoid granting the Supervisor eDirectory right to Server objects because this right is inherited by the file system.

Designing eDirectory for E-Business

If you use eDirectory for e-business, whether you are providing a portal for services or sharing data with another business, the recommendations mentioned in this chapter might not apply to you.

You might want to follow these suggested eDirectory e-business design guidelines.

◆ Create a tree with a limited number of containers.

This guideline depends on the applications you use and your implementation of eDirectory. For example, a global deployment of a messaging server might require the more traditional eDirectory design guidelines discussed earlier in this chapter. Or if you are going to distribute administration of users, you might create a separate Organizational Unit (OU) for each area of administrative responsibility.

◆ Maintain at least two partitions.

Maintain the default partition at the Tree level, and create a partition for the rest of the tree. If you have created separate OUs for administrative purposes, create partitions for each of the OUs.

If you are splitting the load over multiple servers, consider limiting the number of partitions, but still maintain at least two for backup or disaster recovery.

◆ Create at least three replicas of your tree for fault tolerance and load balancing.

Keep in mind that LDAP does not load balance itself. To balance the load on LDAP, consider using Layer 4 switches.

◆ Create a separate tree for e-business. Limit the network resources, such as servers and printers, included in the tree. Consider creating a tree that contains only User objects.

You can use DirXML to link this user tree to your other trees that contain network information. For more information on DirXML, see DirXML Administration Guide (http://www.novell.com/documentation/lg/ndsdirxml/docui/index.html).

◆ Use auxiliary classes to customize your schema.

If a customer or application requires a User object that is different from the standard inetOrgPerson, use auxiliary classes to customize your

schema. Using auxiliary classes allows application designers to change the attributes used in the class without having to recreate the tree.

◆ Increase LDIF-import performance.

When the Novell Import Conversion Export utility is used, eDirectory indexes each object during the process. This can slow down the LDIF-import process. To increase the LDIF-import performance, suspend all indexes from the attributes of the objects you are creating, use the Novell Import Conversion Export utility, then resume indexing the attributes.

◆ Implement globally unique common names (CN).

eDirectory allows the same CN in different containers. However, if you use globally unique CNs you can perform searches on CN without implementing logic for dealing with multiple replies.

Understanding the Novell Certificate Server

Novell Certificate Server™ allows you to mint, issue, and manage digital certificates by creating a Security container object and an Organizational Certificate Authority (CA) object. The Organizational CA object enables secure data transmissions and is required for Web-related products such as NetWare Web Manager and NetWare Enterprise Web Server. The first eDirectory server will automatically create and physically store the Security container object and Organizational CA object for the entire eDirectory tree. Both objects are created and must remain at the top of the eDirectory tree.

Only one Organizational CA object can exist in an eDirectory tree. Once the Organizational CA object is created on a server, it cannot be moved to another server. Deleting and re-creating an Organizational CA object invalidates any certificates associated with the Organizational CA.

IMPORTANT: Make sure that the first eDirectory server is the server that you intend to permanently host the Organizational CA object and that the server will be a reliable, accessible, and continuing part of your network.

If this is not the first eDirectory server on the network, the installation program finds and references the eDirectory server that holds the Organizational CA object. The installation program accesses the Security container and creates a Server Certificate object.

If an Organizational CA object is not available on the network, Web-related products will not function.

On Linux or Solaris, the administrator must manually create an Organizational CA object and the Server Certificate object.

Rights Required to Perform Tasks on Novell Certificate Server

To complete the tasks associated with setting up Novell Certificate Server, the administrator needs to have rights as described in the following table.

Table 19   Rights Required to Perform Tasks on Novell Certificate Server

Task Associated with Setting Up Novell Certificate Server	Rights Required
Base security setup for installing the first server into a new tree, or upgrading the first server in a tree where there is no base security previously installed.	Create rights at the root of the eDirectory tree
	Supervisor rights at the root of the tree
	Supervisor rights on the Security container
Base security setup for installing subsequent servers.	Supervisor rights on the server's container
	Supervisor rights on the W0 object (located inside the Security container)
Creating the Organizational CA	Supervisor rights on the Security container
Creating Server Certificate objects	Supervisor rights on the server's container
	Read rights to the NDSPKI:Private Key attribute on the Organizational CA's object

The root administrator can also delegate the authority to use the Organizational CA by assigning the following rights to subcontainer administrators. Subcontainer administrators require the following rights to install Novell eDirectory with SSL security:

1. Read rights to the NDSPKI:Private Key attribute on the Organizational CA's object, located in the Security container.

2. Supervisor rights to the W0 object located in the Security container, inside the KAP object.

These rights are assigned to a group or a role. All the administrative users are defined in a Group or Role. For a complete list of required rights to perform specific tasks associated with Novell Certificate Server, refer to the online documentation for Novell Certificate Server.

Ensuring Secure eDirectory Operations on Linux and Solaris Systems

eDirectory includes Public Key Cryptography Services (PKCS), which contains the Novell Certificate Server™ that provides Public Key Infrastructure (PKI) services, Novell International Cryptographic Infrastructure (NICI), and SAS*-SSL server.

The following sections provide information about performing secure eDirectory operations:

- "Verifying Whether NICI Is Installed and Initialized on the Server" on page 77
- "Initializing the NICI Module on the Server" on page 78
- "Starting the Certificate Server (PKI Services)" on page 78
- "Stopping the Certificate Server (PKI Services)" on page 78
- "Creating a Certificate Authority" on page 78
- "Creating a Key Material Object" on page 79
- "Exporting a Self-Assigned CA Out of eDirectory in DER Format" on page 80

For information about using external certificate authority, refer to Novell Certificate Server Administration Guide (http://www.novell.com/documentation/lg/crt203ad/docui/index.html).

Verifying Whether NICI Is Installed and Initialized on the Server

Verify the following conditions, which indicate that the NICI module has been properly installed and initialized:

❑ The file **/etc/nici.cfg** exists

❑ The directory **/var/novell/nici** exists

❑ The file **`/var/novell/nici/primenici`** exists

If these conditions are not met, you must initialize the NICI module on the server as explained in .

Initializing the NICI Module on the Server

1 Stop the eDirectory server.

- ◆ On Linux systems type **`/etc/rc.d/init.d/ndsd stop`**
- ◆ On Solaris systems, type **`/etc/init.d/ndsd stop`**

2 Verify whether the NICI package is installed.

- ◆ On Linux systems type **`rpm -qa | grep nici`**
- ◆ On Solaris systems type **`pkginfo | grep NOVLniu0`**

If the NICI package is not installed, install it now.

NOTE: You will not be able to proceed if the NICI package is not installed.

3 Copy the .nfk file provided with the package to the /var/novell/nici directory.

Execute the program **`/var/novell/nici/primenici`**

4 Start the eDirectory server.

- ◆ On Linux systems, type **`/etc/rc.d/init.d/ndsd start`**
- ◆ On Solaris systems, type **`/etc/init.d/ndsd start`**

Starting the Certificate Server (PKI Services)

To start PKI services, enter **`npki -1`**.

Stopping the Certificate Server (PKI Services)

To stop PKI services, enter **`npki -u`**.

Creating a Certificate Authority

1 From ConsoleOne, right-click the security object at the Tree object level > click New > click Object.

2 Select NDSPKI: Certificate Authority > click OK > follow the online instructions.

3 Select the target server > enter an eDirectory object name.

4 In Creation Method, select Custom > click Next.

5 Select the key size > use the default values for other options > click Next.

6 In the Select Certificate Basic Constraints option, use the default values > click Next.

7 In Specify the Certificate Parameters, for Validity Period select Specify Dates.

8 For Effective Date, select a few days (3-5) before the system date > use the default values for all other options.

NOTE: The pkiconfig utility can also be utilized to create a Certificate Authority. For more information on using the pkiconfig utility, refer to the manpage pkiconfig.1m.

Creating a Key Material Object

1 From ConsoleOne, right-click the container the LDAP Server object is in > click New > click Object.

2 Select NDSPKI: Key Material > OK.

3 Select the target server > enter a name > in Creation Method, select Custom > click Next.

4 Use the default values for Specify the Certificate Authority option, which will sign the certificate > click Next.

5 In Specify an RSA Key Size and How the Key Is to Be Used, select an appropriate key size > use the default values for all other options > click Next.

6 In Specify the Certificate Parameters, for Validity Period select Specify Dates.

7 For Effective Date, select a few days (3-5) before the system date > use the default values for all other options > click Next.

8 In Specify the Trusted Root Certificate to Be Associated with Server Certificate, use the default values > click Next.

9 Click Finish to create a key material.

10 In the General Property page, select the SSL certificate (KMO) > click Refresh NLDAP Server Now > click Close.

NOTE: The pkiconfig utility can also be used to create a Key Material Object. For more information on using the pkiconfig utility, refer to the manpage pkiconfig.1m.

Exporting a Self-Assigned CA Out of eDirectory in DER Format

1 Double-click the KMO object > go to the Certificates Property page > select Trusted Root Certificate > click Export > select File in Binary DER format > click OK.

2 Include this file in all command line operations that establish secure connections to eDirectory.

Synchronizing Network Time

Time synchronization is a service that maintains consistent server time across the network. Time synchronization is provided by the server operating system, not by eDirectory. eDirectory maintains its own internal time to ensure the proper order of eDirectory packets, but it gets its time from the server operating system.

This section focuses on integrating NetWare time synchronization with the time synchronization of Windows, Linux, and Solaris.

Synchronizing Time on NetWare Servers

In IP networks and mixed protocol networks, NetWare 5.*x* servers communicate time with other servers using IP. NetWare 5.*x* servers use TIMESYNC.NLM and Network Time Protocol (NTP) to accomplish this.

Time synchronization in NetWare 5.*x* always uses TIMESYNC.NLM, whether servers are using IP only, IPX™ only, or both protocols. TIMESYNC.NLM loads when a server is installed. NTP can be configured through TIMESYNC.NLM.

If your network also uses Windows, Linux, or Solaris, you should use NTP to synchronize the servers because it is a standard to provide time synchronization.

For NetWare 3 and NetWare 4, third-party NTP time services are available.

For more information on time synchronization software, see the U.S. Naval Time Service Department Web site (http://tycho.usno.navy.mil).

NTP

NTP functions as part of the UDP protocol suite, which in turn is part of the TCP/IP protocol suite. Therefore, a computer using NTP must have the TCP/

IP protocol suite loaded. Any computers on your network with Internet access can get time from NTP servers on the Internet.

NTP synchronizes clocks to the Universal Time Coordinated (UTC) standard, the international time standard.

NTP introduces the concept of a stratum. A stratum-1 server has an attached accurate time piece such as a radio clock or an atomic clock. A stratum-2 server gets time from a stratum-1 server, and so on.

For NetWare 5 servers, you can load NTP.NLM to implement NTP time synchronization through TIMESYNC.NLM. When NTP is configured with the TIMESYNC.NLM on an IP server, NTP becomes the time source for both IP and IPX servers. In this case, IPX servers must be set to secondary servers.

For more information on time synchronization, refer to *Network Time Management Administration Guide* in the NetWare 6 documentation set on the Novell Documentation Web site (http://www.novell.com/documentation).

TIMESYNC.NLM

TIMESYNC.NLM synchronizes time among NetWare servers. You can use TIMESYNC.NLM with an external time source like an Internet NTP server. You can also configure Novell Client workstations to update their clocks to servers running the TIMESYNC.NLM.

For more information on time synchronization, refer to *Network Time Management Administration Guide* in the NetWare 6 documentation set on the Novell Documentation Web site (http://www.novell.com/documentation).

Synchronizing Time on Windows Servers

Windows does not include an NTP time-synchronization utility. You can obtain an NTP-compatible timeserver in the *Windows NT 4.0 Resource Kit*.

For more information on time synchronization for Windows, refer to the server documentation.

Synchronizing Time on Linux or Solaris Systems

You can use the TIMESYNC 5.09 utility to synchronize time on Linux, Solaris, and NetWare systems. The TIMESYNC utility is available as part of NetWare 5 Support Pack 2 and can be downloaded from the Novell Support Connection Web page (http://support.novell.com).

1 If xntpd is running on the Linux or Solaris systems, kill the process.

- ◆ On Linux systems, type **/etc/rc.d/init.d/xntpd stop**

- ◆ On Solaris systems, type **/etc/init.d/xntpd stop**

To set up the Linux or Solaris server as a Timesync server in a mixed network of NetWare and Linux or Solaris servers:

1 Modify the ntp.conf file.

- ◆ On Linux systems, enter the following in the /etc/ntp.conf file:

  server *IP_address_of_the_Linux_system*
  fudge *IP_address_of_the_Linux_system* stratum 0

- ◆ On Solaris systems, enter the following in the /etc/inet/ntp.conf file:

  server *IP_address_of_the_Solaris_system*
  fudge *IP_address_of_the_Solaris_system* stratum 0

2 Start xntpd.

- ◆ On Linux systems, type **/etc/rc.d/init.d/xntpd**

- ◆ On Solaris systems, type **/etc/init.d/xntpd**

3 Verify ntptrace.

The following information displays:

```
localhost:stratum1, offset 0.000060. synch distance
  0.01004, refid 'LCL'
```

The stratum number can be any number between 1 and 14.

4 On the NetWare server, load monitor > go to Server Parameters > go to Time > go to Timesync Time Source > enter the following:

- ◆ On Linux systems, type the following:

  ***IP_address_of_the_Linux_system*:123;**

- ◆ On Solaris systems, type the following:

  ***IP_address_of_the_Solaris_system*:123;**

5 Save and quit.

This enables the NetWare sever to synchronize time using NTP.

To set up a Linux or Solaris system as a Timesync client:

1 Enter the following line in /etc/ntp.conf (on Linux systems) or /etc/inet/ntp.conf (on Solaris systems):

```
server IP_address_of_the_Timesync_server
```

2 Use the ntpdate command to adjust the time on the Linux or Solaris machine to be as close to the Timesync server as possible.

3 Repeat the following command until the time is adjusted to the Timesync server:

```
ntpdate IP_address_of_the_Timesync_server
```

4 Start xntpd.

5 Verify ntptrace.

The following information displays after a few minutes:

```
localhost:stratum 2, offset 0.000055, synch distance
  0.02406 Solaris_server_name: stratum 1, offset
  0.000030, synch distance 0.01064, refid 'LCL'
```

The stratum number in the first line can be any number between 2 and 15. If the number is below 16, the machine is synchronized with the machine in the second line.

Verifying Time Synchronization

To verify that time is synchronized in the tree, run DSREPAIR from a server in the Tree that has at least read/write rights to the Tree object.

NetWare

1 At the server console, load DSREPAIR.

2 Select Time Synchronization.

For help interpreting the log, click F1.

Windows

1 Go to the NDSCONSOLE > select DSREPAIR > click Start.

2 Click Repair > Time Synchronization.

Linux and Solaris

1 Run the following command:

```
ndsrepair -T
```

3 Understanding Novell eDirectory

This chapter introduces the concepts and the components that comprise Novell® eDirectory™.

Novell eDirectory

In simplest terms, eDirectory is a list of objects that represent network resources, such as network users, servers, printers, print queues, and applications. Figure 2 shows a few of the objects as viewed in the ConsoleOne™ management utility.

Figure 2   eDirectory Objects in ConsoleOne
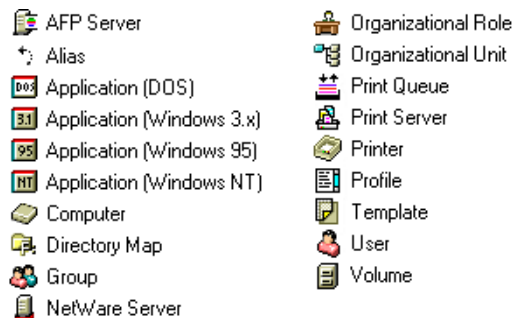
Some object classes might not be available, depending on the actual schema configured on the eDirectory server.

For more information on objects, see "Object Classes and Properties" on page 90.

The directory is physically stored as a set of database files on a server. If the server hosts file system volumes, these files are on volume SYS:. If no volumes are present, the directory is stored on the server's local disk.

If you have more than one eDirectory server on the network, the directory can be replicated on multiple servers.

Ease of Management through ConsoleOne

eDirectory allows for easy, powerful, and flexible management of network resources. It also serves as a repository of user information for groupware and other applications. These applications access your directory through the industry-standard Lightweight Directory Access Protocol (LDAP).

eDirectory ease-of-management features include a powerful tree structure, an integrated management utility, and single login and authentication.

The management console for eDirectory is ConsoleOne which is a 100-percent Java*, directory-enabled framework for running Novell network administration utilities. Management applications are snapped into ConsoleOne, which provides an intuitive graphical interface and a single point of control for all network administration and management functions. The Novell snap-ins to ConsoleOne fully leverage eDirectory to enable role-based administration and greater levels of security.

For more information, refer to ConsoleOne User Guide.

Powerful Tree Structure

eDirectory organizes objects in a tree structure, beginning with the top Tree object, which bears the tree's name.

Whether your eDirectory servers are running NetWare®, UNIX*, or Windows* NT*/2000, all resources can be kept in the same tree. You won't need to access a specific server or domain to create objects, grant rights, change passwords, or manage applications.

The hierarchical structure of the tree gives you great management flexibility and power. These benefits primarily result from two features: container objects and inheritance.

The [Root] object, which was used in earlier versions of eDirectory, has been renamed Tree as shown in .

Figure 3   **Tree Object in ConsoleOne**

Container Objects

Container objects allow you to manage other objects in sets, rather than individually. There are three common classes of container objects as seen in Figure 4:

Figure 4   **Common Classes of Container Objects**
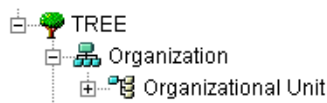
The Tree object is the top container object in the tree. It usually contains your company's Organization object.

Organization is normally the first container class under the Tree object. The Organization object is typically named after your company. Small companies keep management simple by having all other objects directly under the Organization object.

Organizational Unit objects can be created under the Organization to represent distinct geographical regions, network campuses, or individual departments. You can also create Organizational Units under other Organizational Units to further subdivide the tree.

Other classes of container objects are Country and Locality, which are typically used only in multinational networks.

The Domain object is new to eDirectory 8.6 and can be created under the Tree object or under Organization, Organizational Unit, Country, and Locality objects.

You can perform one task on the container object that applies to all objects within the container. Suppose you want to give a user named Amy complete management control over all objects in the Accounting container. See Figure 5 on page 88.

Figure 5   Container Object in ConsoleOne
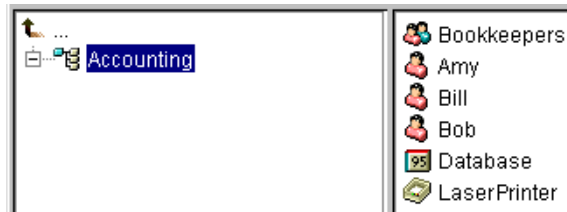
To do this, right-click the Accounting object > select Trustees of This Object > add Amy as a trustee. Next, select the rights you want Amy to have > click OK. Now Amy has rights to manage the Database application, the Bookkeepers group, the LaserPrinter printer, and the users Amy, Bill, and Bob.

Inheritance

Another powerful feature of eDirectory is rights inheritance. Inheritance means that rights flow down to all containers in the tree. This allows you to grant rights with very few rights assignments. For example, suppose you want to grant management rights to the objects shown in Figure 6.

Figure 6   Sample Objects in ConsoleOne
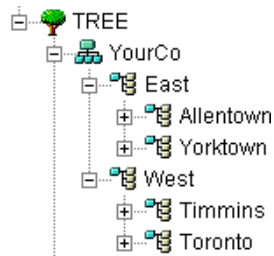
You could make any of the following assignments:

- If you grant a user rights to Allentown, the user can only manage objects in the Allentown container.

- If you grant a user rights to East, the user can manage objects in the East, Allentown, and Yorktown containers.

- If you grant a user rights to YourCo, the user can manage any objects in any of the containers shown.

For more information on assigning rights, see "eDirectory Rights" on page 132.

Integrated Management Utility (ConsoleOne)

ConsoleOne is a utility for managing the entire network. It is like a central console with controls for every aspect of the network.

You can use ConsoleOne on a Windows 95, 98, or NT computer, on a NetWare server, or on a UNIX system to perform the following supervisory tasks:

- Configure LDAP- and XML-based access to eDirectory

- Create objects representing network users, devices, and resources

- Define templates for creating new user accounts

- Find, modify, move, and delete network objects

- Define rights and roles to delegate administrative authority

- Extend the eDirectory schema to allow custom object types and properties

- Partition and replicate the eDirectory database across multiple servers

- Manage files and folders on NetWare volumes

ConsoleOne is an extensible framework that you can use to perform other management functions based on the application snap-ins that have been loaded in to ConsoleOne. For more information, see "Administration Basics" in *ConsoleOne User Guide.*

The following is a list of eDirectory snap-ins to ConsoleOne:

- NDS® Import/Export Wizard
- Novell Certificate Server™
- NDS Administration
- NDS Partition and Replication
- Novell Reporting Services
- Filtered Replica Configuration Wizard
- Service Location Protocol (SLP)
- Index Manager
- Novell LDAP (NLDAP)
- NDS Wanman

Single Login and Authentication

With eDirectory, users log in to a global directory, so you don't need to manage multiple server or domain accounts for each user, and you don't need to manage trust relationships or pass-through authentication among domains.

A security feature of the directory is authentication of users. Before a user logs in, a User object must be created in the directory. The User object has certain properties, such as a name and password.

When the user logs in, eDirectory checks the password against the one stored in the directory for that user and grants access if they match.

Object Classes and Properties

The definition of each type of eDirectory object is called an object class. For instance, User and Organization are object classes. Each class of object has certain properties. A User object, for example, has Login Name, Password, Last Name, and many other properties.

The schema defines the object classes and properties, along with the rules of containment (what containers can contain which objects). eDirectory ships with a base schema that you, or the applications you use, can extend. For more information about schemas, see "Schema" on page 114.

Container objects contain other objects and are used to divide the tree into branches, while leaf objects represent network resources.

List of Objects

Table 20 and Table 21 on page 92 list eDirectory object classes. Added services can create new object classes in eDirectory that are not listed below. Also, all classes might not be available on all server operating systems hosting eDirectory.

Table 20   **eDirectory Container Object Classes**

Container Object (Abbreviation)	Description
Tree	Represents the beginning of your tree. For more information, see "Tree" on page 94.
Country (C)	Designates the countries where your network resides, and organizes other directory objects within the country. For more information, see "Country" on page 97.
License Container (LC)	Created automatically when you install a license certificate or create a metering certificate using Novell Licensing Services (NLS) technology. When an NLS-enabled application is installed, it adds a License Container container object to the tree and a License Certificate leaf object to that container.
Organization (O)	Helps you organize other objects in the directory. The Organization object is a level below the Country object (if you use the Country object). For more information, see "Organization" on page 94.
Organizational Unit (OU)	Helps you to further organize other objects in the directory. The Organizational Unit object is a level below the Organization object. For more information, see "Organizational Unit" on page 96.

Container Object (Abbreviation)	Description
Domain (DC)	Helps you to further organize other objects in the directory. The Domain object can be created under the Tree object or under Organization, Organizational Unit, Country, and Locality objects. For more information, see "Domain" on page 97.

Table 21   **eDirectory Leaf Object Classes**

Leaf Object	Description
AFP Server	Represents an AppleTalk* Filing Protocol server that operates as a node on your eDirectory network. It usually also acts as a NetWare router to, and the AppleTalk server for, several Macintosh* computers.
Alias	Points to the actual location of an object in the directory. Any directory object located in one place in the directory can also appear to be in another place in the directory by using an Alias. For more information, see "Alias" on page 106.
Application	Represents a network application. Application objects simplify administrative tasks such as assigning rights, customizing login scripts, and launching applications.
Computer	Represents a computer on the network.
Directory Map	Refers to a directory in the file system. For more information, see "Directory Map" on page 107.
Group	Assigns a name to a list of User objects in the directory. You can assign rights to the group instead of to each user, then the rights transfer to each user in the group. For more information, see "Group" on page 103.

Leaf Object	Description
License Certificate	Use with NLS technology to install product license certificates as objects in the database. License Certificate objects are added to the Licensed Product container when an NLS-aware application is installed.
Organizational Role	Defines a position or role within an organization.
Print Queue	Represents a network print queue.
Print Server	Represents a network print server.
Printer	Represents a network printing device.
Profile	Represents a login script used by a group of users who need to share common login script commands. The users don't have to be in the same container. For more information, see "Profile" on page 109.
Server	Represents a server running any operating system. For more information, see "Server" on page 99.
Template	Represents standard User object properties that can be applied to new User objects.
User	Represents the people who use your network. For more information, see "User" on page 101.
Unknown	Represents an object for which ConsoleOne has no custom icon.
Volume	Represents a physical volume on the network. For more information, see "Volume" on page 99.

Container Object Classes

Tree

The Tree container, formerly [Root], is created when you first install eDirectory on a server in your network. As the top-most container, it usually holds Organization objects, Country objects, or Alias objects.

What Tree Represents

Tree represents the top of your tree.

Usage

Tree is used to make universal rights assignments. Because of inheritance, any rights assignments you make to Tree as the target apply to all objects in the tree. See "eDirectory Rights" on page 132. The [Public] trustee has the Browse right and Admin has the Supervisor right to Tree by default.

Important Properties

The Tree object has a Name property, which is the tree name you supplied when installing the first server. The tree name is shown in the hierarchy of ConsoleOne.

Organization

An Organization container object is created when you first install eDirectory on a server in your network. As the top-most container under Tree, it usually holds Organizational Unit objects and leaf objects.

The User object named Admin is created by default in your first Organization container.

What an Organization Object Represents

Normally the Organization object represents your company, although you can create additional Organization objects under Tree. This is typically done for networks with distinct geographical districts or for companies with separate eDirectory trees that have merged.

Usage

The way you use Organization objects in your tree depends on the size and structure of your network. If the network is small, you should keep all leaf objects under one Organization object.

For larger networks, you can create Organizational Unit objects under the Organization to make resources easier to locate and manage. For example, you can create Organizational Units for each department or division in your company.

For networks with multiple sites, you should create an Organizational Unit for each site under the Organization object. That way, if you have (or plan to have) enough servers to partition the directory, you can do so logically along site boundaries.

For easy sharing of company-wide resources, such as printers, volumes, or applications, create corresponding Printer, Volume, or Application objects under the Organization.

Important Properties

The most useful properties for Organization are listed below. Only the Name property is required. For a complete list of properties, select an Organization object in ConsoleOne. To display a description for each page of properties, click Help.

- ◆ Name

  Typically, the Name property is the same as your company's name. Of course, you can shorten it for simplicity. For instance, if the name of your company is Your Shoe Company, you might use YourCo.

  The Organization name becomes part of the context for all objects created under it.

- ◆ Login Script

  The Login Script property contains commands that are executed by any User objects directly under the Organization. These commands are run when a user logs in.

Organizational Unit

You can create Organizational Unit (OU) container objects to subdivide the tree. Organizational Units are created with ConsoleOne under an Organization, Country, or another Organizational Unit.

Organizational Units can contain other Organizational Units and leaf objects such as User and Application objects.

What an Organizational Unit Object Represents

Normally the Organizational Unit object represents a department, which holds a set of objects that commonly need access to each other. A typical example is a set of Users, along with the Printers, Volumes, and Applications that those Users need.

At the highest level of Organizational Unit objects, each Organizational Unit can represent each site (separated by WAN links) in the network.

Usage

The way you use Organizational Unit objects in your tree depends on the size and structure of your network. If the network is small, you might not need any Organizational Units.

For larger networks, you can create Organizational Unit objects under the Organization to make resources easier to locate and manage. For example, you can create Organizational Units for each department or division in your company. Remember that administration is easiest when you keep User objects together in the Organizational Unit with the resources they use most frequently.

For networks with multiple sites, you can create an Organizational Unit for each site under the Organization object. That way, if you have (or plan to have) enough servers to partition the directory, you can do so logically along site boundaries.

Important Properties

The most useful properties for the Organizational Unit are listed below. Only the Name property is required. For a complete list of properties, select an Organizational Unit object in ConsoleOne. To display a description for each page of properties, click Help.

◆ Name

Typically, the Name property is the same as the department name. Of course, you can shorten it for simplicity. For instance, if the name of your department is Accounts Payable, you can shorten it to AP.

The Organizational Unit name becomes part of the context for all objects created under it.

◆ Login Script

The Login Script property contains commands that are executed by any User objects directly under the Organizational Unit. These commands are run when a user logs in.

Country

You can create Country objects directly under the Tree object using ConsoleOne. Country objects are optional and only required for connection to certain X.500 global directories.

What a Country Object Represents

The Country object represents the political identity of its branch of the tree.

Usage

Most administrators do not create a Country object, even if the network spans countries, since the Country object only adds an unnecessary level to the tree. You can create one or many Country objects under the Tree object, depending on the multinational nature of your network. Country objects can only contain Organization objects.

If you do not create a Country object and find that you need one later, you can always modify the tree to add one.

Important Properties

The Country object has a two-letter Name property. Country objects are named with a standard two-letter code such as US, UK, or DE.

Domain

You can create Domain objects directly under the Tree object using ConsoleOne. You can also create them under Organization, Organization Unit, Country, and Location objects.

What a Domain Object Represents

The Domain object represent DNS domain components. Domain objects let you use your Domain Name System location of services resource records (DNS SRV) to locate services in your tree.

Using Domain objects, a tree could look something like this:

DS=Novell.DC=Provo.DC=USA

In this example, all subcontainers are domains. You can also use Domain objects in a mixed tree, such as:

DC=Novell.O=Provo.C=USA

Or

OU=Novell.DC=Provo.C=USA

Usually, the top-most Domain is the overall Tree, with subdomains under Tree. For example, machine1.novell.com could be represented by DC=machine1.DC=novell.DC=com in a tree representation.

Domains give you a more generic way to set up an eDirectory tree. If all containers and subcontainers are DC objects, users do not have to remember C, O, or OUs when searching for objects.

Usage

NetWare 4.*x* and 5.*x* trees cannot have Domain objects at the top of the tree. With NetWare 4.*x* and 5.*x*, the NCP Server object can be placed in an Organization, Country, Organizational Unit, or Locality container, but not in a Domain container. With NetWare 6, however, you can place Domain objects at the top of the tree, and you can place the NCP server object in a Domain container.

For older installations of NetWare (such as 4.*x*), when you prepare the tree to install or upgrade NetWare 5 or later, the NDS500.SCH file will automatically run. After the first server is installed into the tree, this file extends the schema to allow the Domain container to be created anywhere and hold most directory objects.

Leaf Object Classes

Server

A Server object is created in the tree automatically whenever you install eDirectory on a server. The object class can be any server running eDirectory.

You can also create a Server object to represent a NetWare 2 or NetWare 3 bindery server.

What a Server Object Represents

The Server object represents a server running eDirectory, or a bindery-based (NetWare 2 or NetWare 3) server.

Usage

The Server object serves as a reference point for replication operations. A Server object that represents a bindery-based server allows you to manage the server's volumes with ConsoleOne.

Important Properties

The Server object has a Network Address property, among others. For a complete list of properties, select a Server object in ConsoleOne. To display a description for each page of properties, click Help.

- Network Address

  The network address property displays the protocol and address number for the server. This is useful for troubleshooting at the packet level.

Volume

When you create a physical volume on a server, a Volume object is automatically created in the tree. By default, the name of the Volume object is the server's name with an underscore and the physical volume's name appended (for example, YOSERVER_SYS).

Volume objects are supported only on NetWare. UNIX file system partitions cannot be managed using Volume objects.

What a Volume Object Represents

A Volume object represents a physical volume on a server, whether it is a writable disk, a CD, or other storage medium. The Volume object in eDirectory does not contain information about the files and directories on that volume, though you can access that information through ConsoleOne. File and directory information is retained in the file system itself.

Usage

In ConsoleOne, click the Volume icon to manage files and directories on that volume. ConsoleOne provides information about the volume's free disk space, directory entry space, and compression statistics.

You can also create Volume objects in the tree for NetWare 2 and NetWare 3 volumes.

Important Properties

In addition to the required Name and Host Volume properties, there are other important Volume properties.

- ◆ Name

  This is the name of the Volume object in the tree. By default, this name is derived from the name of the physical volume, though you can change the object name.

- ◆ Host Server

  This is the server on which the volume resides.

- ◆ Version

  The Version property gives the NetWare or eDirectory version of the server hosting the volume.

- ◆ Host Volume

  This is the physical volume name. Since the actual Volume object name does not need to reflect the physical volume name, this property is necessary to associate the Volume object with the physical volume.

User

A User object is required for logging in. When you install the first server into a tree, a User object named Admin is created. Log in as Admin the first time.

You can use the following methods to create or import User objects:

◆ ConsoleOne

For more information on ConsoleOne, refer to ConsoleOne User Guide.

◆ Replica Advisor in Novell Account Management for Windows NT

For more information on the Replica Advisor, refer to *Account Management Administration Guide*.

◆ Batches from database files

For more information on using batch files, refer to "Designing the eDirectory Tree" on page 58.

◆ NetWare upgrade utilities

For more information on upgrade utilities, including importing users from existing bindery servers, refer to "Designing the eDirectory Tree" on page 58.

What a User Object Represents

A User object represents a person who uses the network.

Usage

You should create User objects for all users who need to use the network. Although you can manage User objects individually, you can save time by:

◆ Using Template objects to set default properties for most User objects. The Template applies automatically to new Users you create (not to already-existing ones).

◆ Creating Group objects to manage sets of Users.

◆ Assigning rights using the container objects as trustees when you want that assignment to apply to all User objects in the container.

◆ Selecting multiple User objects by Shift-clicking or Ctrl-clicking. When you do, you can change property values for all selected User objects.

Important Properties

User objects have over 80 properties. For a complete list of properties, select a User object in ConsoleOne. To display a description for each page of properties, click Help.

The Login Name and Last Name properties are required. These and some of the most useful properties are listed below.

 ◆ Account Expiration Date: This property lets you limit the life of a user account. After the expiration date, the account is locked so the user cannot log in.

 ◆ Account Disabled: This property has a system-generated value that indicates a lock on the account so the user cannot log in. The lock might occur if the account has expired or because the user has given too many incorrect passwords in succession.

 ◆ Force Periodic Password Changes: This property lets you enhance security by requiring the user to change passwords after a specified interval.

 ◆ Group Memberships: This property lists all the Group objects that include the User as a member.

 ◆ Home Directory: The Home Directory property refers to a NetWare volume and file system path for the user's own files. Most administrators like to create such a directory so that a user's working files can be kept on the network.

   The directory referred to in this property can be created automatically when you create the User object.

 ◆ Last Login: This is a system-generated property that lists the date and time that the user last logged in.

 ◆ Last Name: The Last Name property, although required, is not used directly by eDirectory. Applications that take advantage of the eDirectory name base can use this property, along with other identification properties such as Given Name, Title, Location, and Fax Number.

 ◆ Limit Concurrent Connections: This property lets you set the maximum number of sessions a user can have on the network at any given time.

◆ Login Name: This is the name shown in ConsoleOne by the User icon. It is also the name supplied by the user when logging in.

eDirectory does not require that login names be unique throughout the network, only in each container. However, you might want to keep login names unique across the company to simplify administration.

Typically, login names are a combination of first and last names, such as STEVET or STHOMAS for Steve Thomas.

◆ Login Script: The Login Script property lets you create specific login commands for a User object. When a user logs in, the container login script runs first. Then a profile login script runs if the User object has been added to the membership list of a Profile object. Finally, the user login script runs (if one exists).

You should put most of the login commands in container login scripts to save administrative time. The user login script can be edited to manage unique exceptions to common needs.

◆ Login Time Restrictions: This property lets you set times and days when the user can log in.

◆ Network Addresses: This property contains system-generated values that list all the IPX™ and/or IP addresses from which the user is logged in. These values are useful for troubleshooting network problems at the packet level.

◆ Require a Password: This property lets you control whether the user must use a password. Other related properties let you set common password constraints such as password length.

◆ Rights to Files and Directories: This property lists all rights assignments made for this user to the NetWare file system. Using ConsoleOne, you can also check a user's effective rights to files and directories, which include those inherited from other objects.

Group

You can create Group objects to help you manage sets of User objects.

What a Group Object Represents

A Group object represents a set of User objects.

Usage

While container objects let you manage all User objects in that container, Group objects are for subsets within a container or in multiple containers.

Group objects have two main purposes:

- ◆ They allow you to grant rights to a number of User objects at once.
- ◆ They allow you to specify login script commands using the IF MEMBER OF syntax.

Static Groups

Static groups identify the member objects explicitly. Each member is assigned to the group explicitly.

Dynamic Groups

Dynamic groups use an LDAP URL to define a set of rules, which when matched by eDirectory User objects, define the members of the group. Dynamic group members share a common set of attributes as defined by the filter specified in the URL.

Dynamic groups let you specify the criteria to be used for evaluating membership in a group. The actual members of the group are evaluated dynamically by eDirectory, which lets you define the group members in terms of a logical grouping, and lets eDirectory automatically add and remove group members. This solution is more scalable and reduces administrative costs, and can supplement normal groups in LDAP to provide increased flexibility.

eDirectory 8.6 lets you create a dynamic group when you want to group users automatically based on any attribute, or when you want to apply ACLs to specific groups which contain matching DNs. For example, you can create a group that automatically includes any DN that contains the attribute Department=Marketing. If you apply a search filter for Department=Marketing, the search returns a group including all DNs containing the attribute Department=Marketing. You can then define a dynamic group from the search results based on this filter. Any User added to the directory who matches the Department=Marketing criteria is automatically added to the group. Any User whose Department is changed to another value (or who is removed from the directory) is automatically removed from the group.

Dynamic groups are created in eDirectory by creating an object of type objectclass="dynamicGroup". A static group object can be converted into a dynamic group by associating an auxiliary class, dynamicGroupAux, to the group object. The dynamic group has the memberQueryURL attribute associated with it.

A "dgIdentity" attribute can be set on the dynamic group object to the distinguished name of an entry, whose credentials and rights should be used to expand the dynamic members of the group.

The groups are managed using the memberQueryURL. A typical memberQueryURL has a base DN, a scope, a filter, and an optional extension.The base DN specifies the search base. Scope specifies the levels below the base to search, and filter is the search filter based on which entries are selected from within the specified scope.

NOTE: To address exceptions to the listing created by the memberQueryURL, dynamic groups also allow for explicit inclusion and exclusion of users.

In eDirectory 8.6, Dynamic Groups cannot be managed through ConsoleOne. Use LDAP commands to manage such groups. The most useful properties associated with Dynamic Groups are dgIdentity and memberQueryURL.

Important Properties

The most useful properties of the Group object are Members and Rights to Files and Directories. For a complete list of properties, select a Group object in ConsoleOne. To display a description for each page of properties, click Help.

 - dgIdentity

    This property holds the DN whose identity the Dynamic Group will use for authentication while searching. The identity must be on the same server as the Dynamic Group.

 - memberQueryURL

    This property defines the set of rules that match with the attributes of the group members.

 - Members

    This property lists all objects in the group. Rights assignments made to the Group object apply to all members of that group.

Alias

You can create an Alias object that points to another object in the tree. Alias objects give users a local name for an object that lies outside their container.

When you rename a container, you have the option of creating an Alias in the former container's place that points to the new name. Workstations and login script commands that reference objects in the container can still access the objects without having the container name updated.

What an Alias Object Represents

An Alias object represents another object, which can be a container, User object, or any other object in the tree. An Alias object does not carry trustee rights of its own. Any trustee authority you grant to the Alias object applies to the object it represents. The Alias can be a target of a trustee assignment, however.

Usage

Create an Alias object to make name resolution easier. Since object naming is simplest for objects in the current context, you should create Alias objects there that point to any resources outside the current context.

For example, suppose users log in and establish a current context in the South container as shown in Figure 7, but need access to the Print Queue object named ColorQ in the North container.

Figure 7   Sample Containers in ConsoleOne
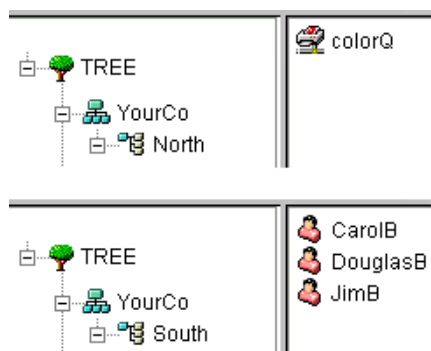
You can create an Alias object in the South container. See Figure 8 on page 107.

Figure 8   Alias Object in ConsoleOne Container
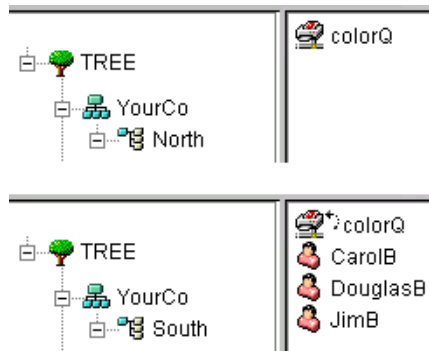
The Alias object points to the original ColorQ object, so setting up printing for the users involves a local object.

Important Properties

Alias objects have an Aliased Object property, which associates the Alias object with the original object.

Directory Map

The Directory Map object is a pointer to a path in the server file system. It allows you to make simpler references to directories.

If your network has no NetWare volumes, you cannot create Directory Map objects.

What a Directory Map Object Represents

A Directory Map object represents a directory on a NetWare volume. (An Alias object, on the other hand, represents an object.)

Usage

Create a Directory Map object to make drive mapping simpler, particularly in login scripts. Using a Directory Map object allows you to reduce complex file system paths to a single name.

Also, when you change the location of a file, you don't need to change login scripts and batch files to reference the new location. You only need to edit the Directory Map object. For example, suppose you were editing the login script for the container South, shown in .

Figure 9   Sample ConsoleOne Container
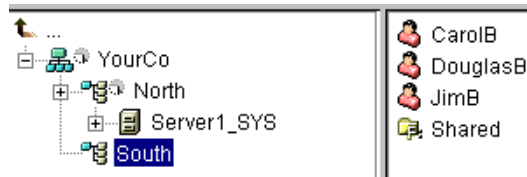
A command mapping drives to the Shared directory on volume SYS: would look like the following:

MAP N:=SYS.North.:Shared

If you created the Shared Directory Map object, the map command would be much simpler:

MAP N:=Shared

Important Properties

The Directory Map object has Name, Volume, and Path properties.

- Name

  The Name property identifies the object in the directory (for example, Shared) and is used in MAP commands.

- Volume

  The Volume property contains the name of the Volume object that the Directory Map object references, such as Sys.North.YourCo.

- Path

  The Path property specifies the directory as a path from the root of the volume, such as PUBLIC\WINNT\NLS\ENGLISH.

Profile

Profile objects help you manage login scripts.

What a Profile Object Represents

A Profile object represents a login script that runs after the container login script and before the user login script.

Usage

Create a Profile object if you want login script commands to run for only selected users. The User objects can exist in the same container or be in different containers. Once you have created the Profile object, you add the commands to its Login Script property. Then make the User objects trustees of the Profile object and add the Profile object to their Profile Membership property.

Important Properties

The Profile object has two important properties: Login Script and Rights to Files and Directories.

 ◆ Login Script

   The Login Script property contains the commands you want to run for users of the Profile.

 ◆ Rights to Files and Directories

   If you have INCLUDE statements in the login script, you need to give the Profile object rights to the files included with the Rights to Files and Directories property.

Context and Naming

The context of an object is its position in the tree. It is nearly equivalent to a DNS domain.

You can see in the following figure that User Bob is in Organizational Unit Accounts, which is in Organizational Unit Finance, which is in Organization YourCo. See Figure 10.

Figure 10   Sample ConsoleOne Container
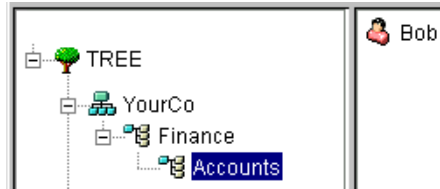
Sometimes, however, you need to express the context of an object in an eDirectory utility. For example, you could be setting up Bob's workstation and need to supply a name context, as shown in Figure 11.

Figure 11   Novell Client NDS Page
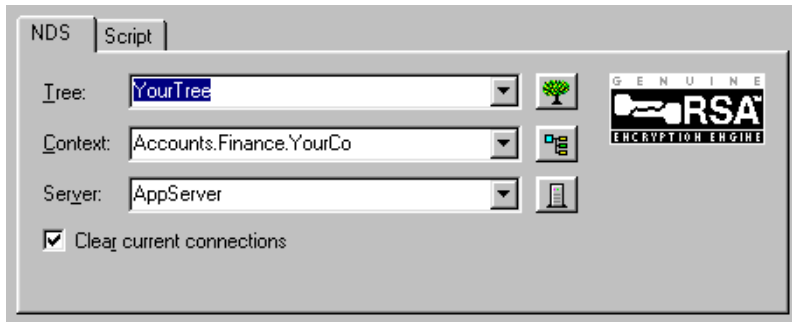
The context is specified as a list of containers separated by periods, between the object in question and the top of the Tree. In the example above, User object Bob is in the container Accounts, which is in the container Finance, which is in the container YourCo.

Distinguished Name

The distinguished name of an object is its object name with the context appended. For example, the complete name of User object Bob is Bob.Accounts.Finance.YourCo.

Typeful Name

Sometimes typeful names are displayed in eDirectory utilities. Typeful names include the object type abbreviations in Table 22.

Table 22   eDirectory Object Type Abbreviations

Object Class	Type	Abbreviation
All leaf object classes	Common Name	CN
Organization	Organization	O
Organizational Unit	Organizational Unit	OU
Country	Country	C
Locality	Locality or State/ Province	L or S

In creating a typeful name, eDirectory uses the type abbreviation, an equal sign, and the object's name. For instance, Bob's partial typeful name is CN=Bob. Bob's complete typeful name is CN=Bob.OU=Accounts.OU=Finance.O=YourCo. You can use typeful names interchangeably with typeless names in eDirectory utilities.

Name Resolution

The process eDirectory uses to find an object's location in the directory tree is called name resolution. When you use object names in eDirectory utilities, eDirectory resolves the names relative to either the current context or the top of the tree.

Current Workstation Context

Workstations have a context set when the networking software runs. This context relatively identifies the location of the workstation in the network. For instance, Bob's workstation would be set to the current context as follows:

Accounts.Finance.YourCo

Current context is a key to understanding the use of leading periods, relative naming, and trailing periods.

Leading Period

Use a leading period to resolve the name from the top of the tree, no matter where the current context is set. In the example below, the leading period tells the CX (Change Context) utility to resolve the name relative to the top of the tree.

CX .Finance.YourCo

eDirectory interprets the command as "Change the context to the Finance container, which is in the YourCo container, resolved from the top of the tree."

The workstation's current context changes to the Finance container, which is in the YourCo container.

Relative Naming

Relative naming means that names are resolved relative to the workstation's current context, rather than the top of the tree. Relative naming never involves a leading period, since a leading period indicates resolution from the top of the tree.

Suppose a workstation's current context is set to Finance. See Figure 12.

Figure 12   Sample ConsoleOne Container
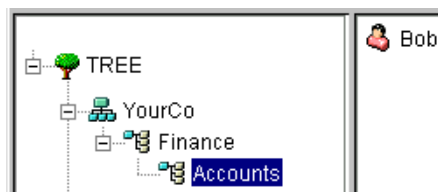
The relative object name of Bob is:

```
Bob.Accounts
```

eDirectory interprets the name as "Bob, which is in Accounts, resolved from the current context, which is Finance."

Trailing Periods

Trailing periods can only be used in relative naming. Therefore, you can't use both a leading period and a trailing period. A trailing period changes the container from which eDirectory resolves the name.

Each trailing period changes the resolution point one container toward the top of the tree. Suppose you want to change your workstation's current context from Timmins to Allentown in the example in .

Figure 13   Sample ConsoleOne Container

```
白-🌳 TREE
    白-🖧 YourCo
        白-🖳 East
            白-🖳 Allentown
            白-🖳 Yorktown
        白-🖳 West
            白-🖳 Timmins
            白-🖳 Toronto
```
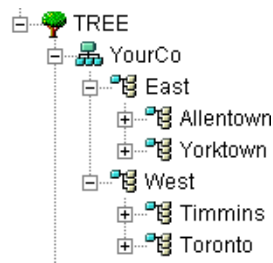
The proper CX command uses relative naming with trailing periods:

```
CX Allentown.East..
```

eDirectory interprets the command as "Change the context to Allentown, which is in East, resolved from two containers up the tree from the current context."

Similarly, if Bob is in the Allentown container and your workstation's current context is Timmins, Bob's relative name is:

```
Bob.Allentown.East..
```

Context and Naming on UNIX

When UNIX user accounts are migrated to eDirectory, the eDirectory context is not used to name users. The context of the user is determined by the UAM component.

Schema

The schema defines the types of objects that can be created in your tree (such as Users, Printers, and Groups) and what information is required or optional at the time the object is created. Every object has a defined schema class for that type of object.

The schema that originally shipped with the product is called the base schema. Once the base schema has been modified in any way–such as adding a new class or a new attribute–then it is considered the extended schema.

You don't have to extend the schema, but you have the ability to do so. The Schema Manager tool in ConsoleOne lets you extend the schema to meet organizational needs. For example, you might want to extend your schema if your organization requires special footwear for employees and you need to keep track of employee shoe sizes. You might want to create a new attribute called Shoe Size and then add it to the User class.

For more information, see .

Schema Manager

Schema Manager is a tool in ConsoleOne. It allows users who have Supervisor rights to a tree to customize the schema of that tree. Schema Manager is accessed from the Tools menu in ConsoleOne after you select a tree.

Use Schema Manager to:

- ◆ View a list of all classes and attributes in the schema.
- ◆ View information on an attribute such as its syntax and flags.
- ◆ Extend the schema by adding a class or an attribute to the existing schema.
- ◆ Create a class by naming it and specifying attributes, flags, containers to which it can be added, and parent classes that it can inherit attributes from.
- ◆ Create an attribute by naming it and specifying its syntax and flags.
- ◆ Add an optional attribute to an existing class.
- ◆ Delete a class or attribute that is not used or that is obsolete.

Schema Classes, Attributes, and Syntaxes

Classes

A class is like a template for a directory object. A directory object is a class that has been filled in with data. In other words:

CLASS + DATA = DIRECTORY OBJECT

Each class has a class name, an inheritance class (unless it is at the top of the class hierarchy), class flags, and a group of attributes. Classes are named like directory objects (User, Printer, Queue, Server, and so on), yet they are just structure, no content.

An inheritance class is a class that is a starting point for defining other object classes. All of the attributes of the inheritance class are inherited by the classes that come below it in the class hierarchy.

A class hierarchy shows how a class is associated with its parent classes. This is a way of associating similar classes and allowing attributes to be inherited. It also defines the types of containers the class is valid in.

When creating a new class, you can use the class hierarchy and the additional attributes available to customize each class. You can specify an inheritance class (which allows the new class to inherit all of the attributes and flags of a class higher in the hierarchy) and then customize the new class by selecting one or more attributes to add to those that were inherited. The additional attributes can be selected as mandatory, naming, or optional attributes.

You can also modify existing classes by adding optional attributes.

Attributes

Attributes are the data fields in the eDirectory database. For example, if a class is like a form, then an attribute is one field on the form. When an attribute is created, it is named (such as *surname* or *employee number*) and given a syntax type (such as *string* or *number*). From then on, it is available in the attribute lists in Schema Manager.

Syntaxes

There are several syntax options from which to choose. These are used to specify the type of data entered for each attribute. The syntax can only be specified when an attribute is created. You cannot modify it later. Available syntaxes include:

- Backlink

  Used to keep track of other servers referring to an object. It is used for internal eDirectory management purposes.

- Boolean

  Used by attributes whose values are True (represented as 1) or False (represented as 0). The single valued flag is set for this syntax type.

- Case Exact String

  Used by attributes whose values are Unicode* strings that are case-sensitive in comparison operations. Two Case Exact Strings match when they are of the same length and their corresponding characters, including case, are identical.

- Case Ignore List

  Used by attributes whose values are ordered sequences of Unicode strings that are case-insensitive in comparisons operations. Two Case Ignore Lists match if the number of strings in each is the same and all corresponding strings match (that is, they are the same length and their corresponding characters are identical).

- Case Ignore String

  Used by attributes whose values are Unicode strings that are not case-sensitive in comparison operations. Two Case Ignore Strings match when they are of the same length and their corresponding characters are identical in all respects except that of case.

- Class Name

  Used by attributes whose values are object class names. Two Class Names match when they are of the same length and their corresponding characters are identical in all respects except that of case.

- Counter

  Used by attributes whose values are incrementally modified numeric signed integers. Any attribute defined using Counter is a single-valued attribute. This syntax differs from Integer in that any value added to an attribute of this syntax is arithmetically added to the total, and any value deleted is arithmetically subtracted from the total.

- Distinguished Name

  Used by attributes whose values are the names of objects in the eDirectory tree. Distinguished Names (DN) are not case-sensitive, even if one of the naming attributes is case-sensitive.

- Email Address

  Used by attributes whose values are strings of binary information. eDirectory makes no assumption about the internal structure of the content of this syntax.

- Facsimile Telephone Number

  Specifies a string that complies with the E.123 standard for storing international telephone numbers and an optional bit string formatted according to recommendation T.20. Facsimile Telephone Number values match when they are of the same length and their corresponding characters are identical, except that all spaces and hyphen characters are ignored during comparison.

- Hold

  Used by attributes that are accounting quantities, whose values are signed integers. This syntax is an accounting quantity (which is an amount tentatively held against a subject's credit limit, pending completion of a transaction). The hold amount is treated similarly to the Counter syntax, with new values added to or subtracted from the base total. If the evaluated hold amount goes to 0, the Hold record is deleted.

- Integer

  Used by attributes represented as signed numeric values. Two Integer values match if they are identical. The comparison for ordering uses signed integer rules.

- Interval

  Used by attributes whose values are signed numeric integers and represent intervals of time. The Interval syntax uses the same representation as the Integer syntax. The Interval value is the number of seconds in a time interval.

- Net Address

  Represents a network layer address in the server environment. The address is in binary format. For two values of Net Address to match, the type, length, and value of the address must match.

- Numeric String

  Used by attributes whose values are numerical strings as defined in the CCITT X.208 definition of Numeric String. For two Numeric Strings to match, the strings must be the same length and their corresponding characters must be identical. Digits (0...9) and space characters are the only valid characters in the numeric string character set.

◆ Object ACL

Used by attributes whose values represent Access Control List (ACL) entries. An Object ACL value can protect either an object or an attribute.

◆ Octet List

Describes an ordered sequence of strings of binary information or Octet String. An Octet List matches a stored list if it is a subset of the stored list. For two Octet Lists to match, they are compared using the same methods as Octet Strings.

◆ Octet String

Used by attributes whose values are strings of binary information not interpreted by eDirectory. These octet strings are non-Unicode strings. For two octet strings to match, they must be the same length, and the corresponding bit sequence (octet) must be identical.

◆ Path

Attributes that represent a file system path contain all the information to locate a file on a server. Two paths match when they are of the same length and their corresponding characters, including case, are identical.

◆ Postal Address

Used by attributes whose values are Unicode strings of Postal Addresses. An attribute value for Postal Address is typically composed of selected attributes from the MHS Unformatted Postal O/R Address Specification version 1 according to recommendation F.401. The value is limited to six lines of 30 characters each, including a postal country name. Two postal addresses match if the number of strings in each is the same and all corresponding strings match (that is, they are the same length and their corresponding characters are identical).

◆ Printable String

Used by attributes whose values are printable strings, as defined in CCITT X.208. The printable character set consists of the following:

  ◆ Uppercase and lowercase alphabetic characters

  ◆ Digits (0...9)

  ◆ Space character

  ◆ Apostrophe (')

  ◆ Left and right parentheses ( )

- Plus sign (+)
- Comma (,)
- Hyphen (-)
- Period (.)
- Forward slash (/)
- Colon (:)
- Equals sign (=)
- Question mark (?)

Two printable strings are equal when they are the same length and their corresponding characters are the same. Case is significant.

- Replica Pointer

  Used by attributes whose values represent partition replicas. A partition of an eDirectory tree can have replicas on different servers. The syntax has six components:

  - Server Name
  - Replica Type (master, secondary, read-only, subordinate reference)
  - Replica Number
  - Replica Root ID
  - Number of Address
  - Address Record

- Stream

  Represents arbitrary binary information. The Stream syntax provides a way to make an eDirectory attribute out of a file on a file server. Login scripts and other stream attributes use this syntax. The data stored in a stream file has no syntax enforcement of any kind. It is purely arbitrary data, defined by the application that created and uses it.

- Telephone Number

  Used by attributes whose values are telephone numbers. The length of telephone number strings must be between 1 and 32 characters. Two telephone numbers match when they are of the same length and their corresponding characters are identical, except that all spaces and hyphen characters are ignored during comparison.

◆ Time

Used by attributes whose values are unsigned integers and represent time expressed in seconds.

◆ Timestamp

Used by attributes whose values mark the time when a particular event occurred. When a significant event occurs, an eDirectory server mints a new Timestamp value and associates the value with the event. Every Timestamp value is unique within an eDirectory partition. This provides a total ordering of events occurring on all servers holding replicas of a partition.

◆ Typed Name

Used by attributes whose values represent a level and an interval associated with an object. This syntax names an eDirectory object and attaches two numeric values to it:

◆ Level of the attribute indicative of its priority

◆ Interval representing the number of seconds between certain events or the frequency of the reference

◆ Unknown

Used by attributes whose attribute definition has been deleted from the schema. This syntax represents strings of binary information.

Understanding Mandatory and Optional Attributes

Every object has a schema class that has been defined for that type of object, and a class is a group of attributes organized in a meaningful way. Some of these attributes are mandatory and some are optional.

Mandatory Attributes

A mandatory attribute is one that must be filled in when an object is being created. For example, if a new user is being created using the User class, which has the employee number as a mandatory attribute, then the new User object cannot be created without providing the employee number.

Optional Attributes

An optional attribute is one that can be filled in if desired but can be left without content. For instance, if a new User object is being created using the User class, which has Other Names as an optional attribute, then the new User object can be created with or without data provided for that attribute, depending on whether the new user is known by other names.

An exception to the rule is when an optional attribute is used for naming, the attribute then becomes mandatory.

Sample Schema

Figure 14 on page 121 is a sample of part of a schema. Your base schema might appear similar.

This icon is assigned to all classes that are extensions to the base schema.

Figure 14   Schema Manager Dialog Box in ConsoleOne
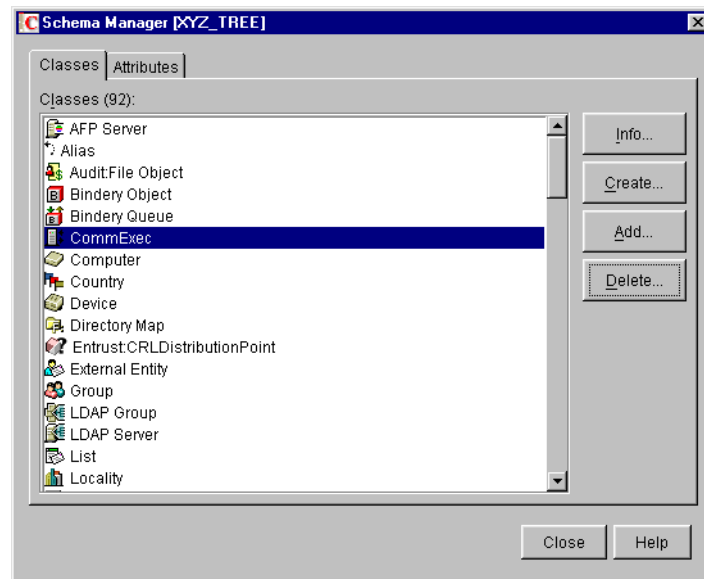
Design the Schema

Designing your schema initially can save you time and effort in the long run. You can view the base schema and determine if it will meet your needs or if modifications are required. If changes are needed, use Schema Manager to extend the schema. See "Extending the Schema" on page 148 for more information.

Partitions

If you have slow or unreliable WAN links or your directory has so many objects that the server is overwhelmed and access is slow, you should consider partitioning the directory. For a complete discussion of partitions, see Chapter 6, "Managing Partitions and Replicas," on page 161.

Partitioning allows you to take part of the directory off one server and put it on another server.

A partition is a logical division of the eDirectory database. A directory partition forms a distinct unit of data in the tree that stores directory information.

Each directory partition consists of a set of container objects, all the objects contained in them, and data about those objects. eDirectory partitions don't include any information about the file system or the directories and files contained there.

Partitioning is done with ConsoleOne. Partitions are identified in ConsoleOne by the following partition icon: 📌. See Figure 15.

Figure 15   Partition and Replica view in ConsoleOne
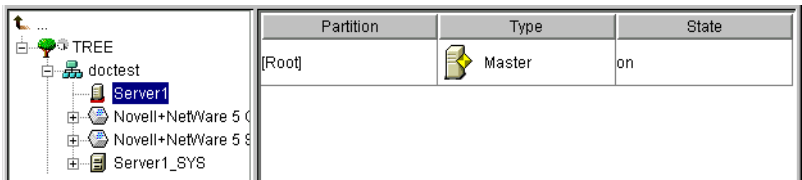
In the example, the partition icon is next to the Tree object. This means it is the top-most container in the partition. No partitions are shown by any other containers, so this partition is the only one.

This is the default partitioning for eDirectory, keeping the entire directory together in one partition.

Notice in the example that Server1 is selected. When you select a server in ConsoleOne and display the Partition and Replica view, any replicas held on that server are shown on the right. In this case, Server1 holds a replica of the only partition. See "Replicas" on page 126.

Partitions

Partitions are named by their top-most container. In Figure 16 on page 123 there are two partitions, named Tree and Finance. YourCo is called a child partition of Tree, since it was split off from the Tree. Tree is called the parent partition of Finance.

Figure 16   Partition and Replica View in ConsoleOne
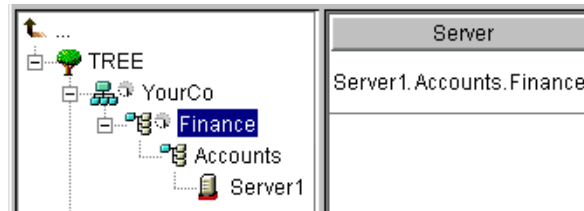
You might create such a partition because the directory has so many objects that the server is overwhelmed and access to eDirectory is slow. Creating the new partition allows you to split the database and pass the objects in that branch to a different server.

Distributing Replicas for Performance

In the preceding example, suppose that Server1 holds replicas of both the Tree partition and the Finance partition. At this point, you haven't gained any performance advantage from eDirectory since Server1 still holds the entire directory (replicas of both partitions).

To gain the desired performance advantage, you need to move one of the replicas to a different server. For instance, if you move the Tree partition to Server2, then Server2 holds all objects in the Tree and YourCo containers. Server1 only holds objects in the Finance and Accounts containers. The load on both Server1 and Server2 is less than it would be with no partitioning.

Partitions and WAN Links

Suppose your network spans two sites, a North site and a South Site, separated by a WAN link. Three servers are at each site. See Figure 17 on page 124.

Figure 17   Sample Containers in ConsoleOne
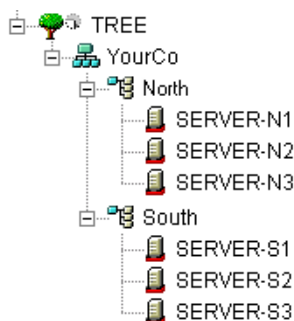
eDirectory performs faster and more reliably in this scenario if the directory is divided in two partitions.

With a single partition, the replicas are either kept at one site or distributed between the two sites. This proves unwieldy for two reasons:

◆ If all replicas are kept on servers at the North site, for instance, users at the South site encounter delays when logging in or accessing resources. If the link goes down, users at the South site can't log in or access resources at all.

◆ If replicas are distributed between sites, users can access the directory locally. However, server-to-server synchronization of replicas happens over the WAN link, so there can be eDirectory errors if the link is unreliable. Any changes to the directory are slow to propagate across the WAN link.

The two-partition solution shown below solves performance and reliability problems over the WAN link. See Figure 18 on page 125.

Figure 18   Sample Partitions in ConsoleOne
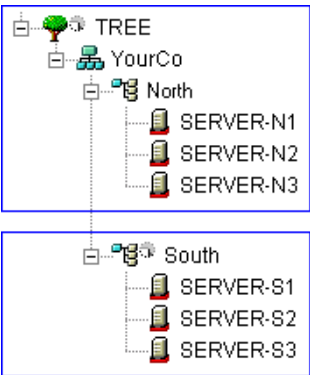
Replicas of the Tree partition are kept on servers at the North site. Replicas of the South partition are kept on servers at the South site, as shown in Figure 19.

Figure 19   Sample Partitions, Severs, and Replica in ConsoleOne

Partition	Server	Replica Type
TREE	SERVER-N1	Master
	SERVER-N2	Read/write
	SERVER-N3	Read/write
South	SERVER-S1	Master
	SERVER-S2	Read/write
	SERVER-S3	Read/write

For each site, the objects that represent local resources are kept locally. Synchronization traffic among servers also happens locally over the LAN, rather than over the slow, unreliable WAN link.

eDirectory traffic is generated over the WAN link, however, when a user or administrator accesses objects at a different site.

Replicas

If you have more than one eDirectory server on your network, you can keep multiple replicas (copies) of the directory. That way, if one server or a network link to it fails, users can still log in and use the remaining network resources. See Figure 20. For a complete discussion of replicas, see Chapter 6, "Managing Partitions and Replicas," on page 161.

Figure 20   eDirectory Replicas
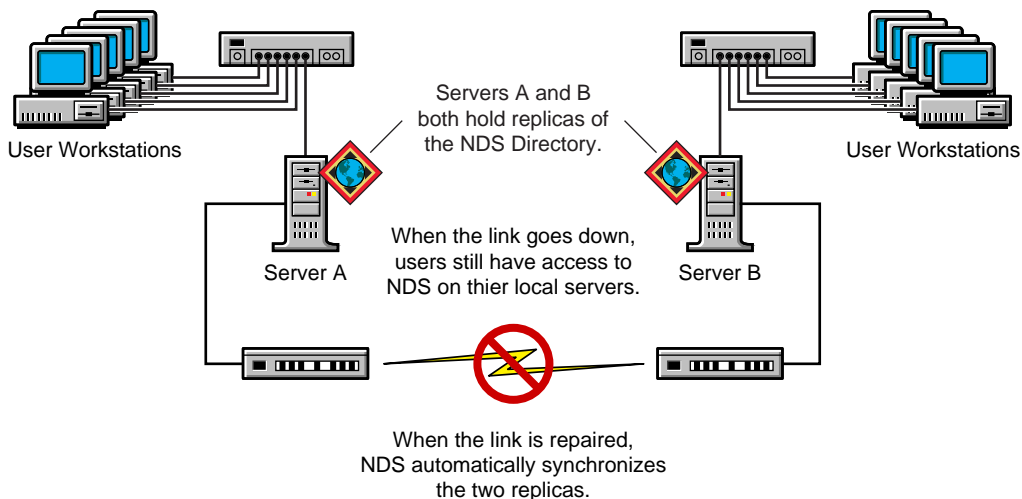
We recommend that you keep three replicas for fault-tolerance of eDirectory (assuming you have three eDirectory servers to store them on). A single server can hold replicas of multiple partitions.

A replica server is a dedicated server that only stores eDirectory replicas. This type of server is sometimes referred to as a DSMASTER server. This configuration is popular with some companies that use many single server remote offices. The replica server provides a place for you to store additional replicas for the partition of a remote office location.

eDirectory replication does not provide fault tolerance for the server file system. Only information about eDirectory objects is replicated. You can get fault tolerance for file systems by using the Transaction Tracking System™ (TTS™), disk mirroring/duplexing, RAID, or Novell Replication Services (NRS).

A master or read/write replica is required on NetWare servers that provide bindery services.

If users regularly access eDirectory information across a WAN link, you can decrease access time and WAN traffic by placing a replica containing the needed information on a server that users can access locally.

The same is true to a lesser extent on a LAN. Distributing replicas among servers on the network means information is usually retrieved from the nearest available server.

Replica Types

eDirectory supports the types of replicas in Figure 21:

Figure 21   Replica Types

| Master | Read/write | Filtered read/write replica |
| Subordinate reference | Read only | Filtered read only replica |
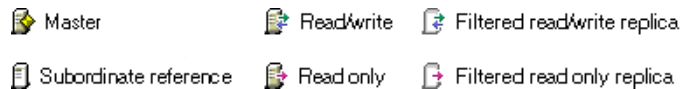
- ◆ "Master Replica" on page 127
- ◆ "Read/Write Replica" on page 128
- ◆ "Read-Only Replica" on page 128
- ◆ "Filtered Read/Write Replica" on page 128
- ◆ "Filtered Read-Only Replica" on page 128
- ◆ "Subordinate Reference Replica" on page 129

Master Replica

By default, the first eDirectory server on your network holds the master replica. There is only one master replica for each partition at a time. If other replicas are created, they are read/write replicas by default. For more information on partitions, see "Partitions" on page 122.

If you're going to bring down the server holding a master replica for longer than a day or two, you can make one of the read/write replicas the master. The original master replica automatically becomes read/write.

A master replica must be available on the network for eDirectory to perform operations such as creating a new replica or creating a new partition.

Read/Write Replica

eDirectory can access and change object information in a read/write replica as well as the master replica. All changes are then automatically propagated to all replicas.

If eDirectory responds slowly to users because of delays in the network infrastructure (such as slow WAN links or busy routers), you can create a read/write replica closer to the users who need it. You can have as many read/write replicas as you have servers to hold them, although more replicas cause more traffic to keep them synchronized with each other.

Read-Only Replica

Read-only replicas receive synchronization updates from master and read/write replicas but don't receive changes directly from clients.

Filtered Read/Write Replica

Filtered read/write replicas contain a filtered set of objects or object classes along with a filtered set of attributes and values for those objects. The contents are limited to the types of eDirectory objects and properties specific in the host server's replication filter. Users can read and modify the contents of the replica, and eDirectory can access and change selected object information. The selected changes are then automatically propagated to all replicas.

With filtered replicas, you can have only one filter per server. This means that any filter defined for a server applies to all filtered replicas on that server. You can, however, have as many filtered replicas as you have servers to hold them, although more replicas cause more traffic to keep them synchronized.

For more information, see .

Filtered Read-Only Replica

Filtered read-only replicas contain a filtered set of objects or object classes along with a filtered set of attributes and values for those objects. They receive synchronization updates from master and read/write replicas but don't receive changes directly from clients. Users can read but not modify the contents of the replica. The contents are limited to the types of eDirectory objects and properties specific in the host server's replication filter.

For more information, see .

Subordinate Reference Replica

Subordinate reference replicas are system-generated replicas that don't contain all the object data of a master or a read/write replica. Subordinate reference replicas therefore don't provide fault tolerance. They are internal pointers that are generated to contain enough information for eDirectory to resolve object names across partition boundaries.

You can't delete a subordinate reference replica; eDirectory deletes it automatically when it is not needed. Subordinate reference replicas are only created on servers that hold a replica of a parent partition but no replicas of its child partitions.

If a replica of the child partition is copied to a server holding the replica of the parent, the subordinate reference replica is automatically deleted.

Filtered Replicas

Filtered replicas contain a filtered set of objects or object classes along with a filtered set of attributes and values for those objects. For example, you might want to create a set of filtered replicas on a single server that contain only User objects from various partitions in the eDirectory tree. In addition to this, you can choose to include only a subset of the User objects' data (for example,. Given Name, Surname, and Telephone Number).

A filtered replica can construct a view of eDirectory data onto a single server. To do this, filtered replicas let you create a scope and a filter. This results in an eDirectory server that can house a well-defined data set from many partitions in the tree.

The descriptions of the server's scope and data filters are stored in eDirectory and can be managed through the Server object in ConsoleOne.

A server hosting one of more filtered replicas has only a single replication filter. Therefore, all filtered replicas on the server contain the same subset of information from their respective partitions. The master partition replica of a filtered replica must be hosted on an eDirectory server running eDirectory eDirectory 8.5 or later.

Filtered replicas can:

- ◆ Reduce synchronization traffic to the server by reducing the amount of data that must be replicated from other servers.

- ◆ Reduce the number of events that must be filtered by DirXML™.

For more information on DirXML, see DirXML Administration Guide (http://www.novell.com/documentation/lg/ndsdirxml/docui/index.html).

◆ Reduce the size of the directory database.

Each replica adds to the size of the database. By creating a filtered replica that contains only specific classes (instead of creating a full replica), you can reduce the size of your local database.

For example, if your tree contains ten thousand objects but only a small percentage of those objects are Users, you could create a filtered replica containing only the User objects instead of a full replica containing all ten thousand objects.

Other than the ability to filter data stored in a local database, the filtered replica is like a normal eDirectory replica and it can be changed back to a full replica at any time. For more information on setting up and managing filtered replicas, see "Setting Up and Managing Filtered Replicas" on page 169.

NOTE: Filtered replicas by default will have the Organization and the Organizational Unit as mandatory filters.

NetWare Bindery Emulation

Many applications, such as print servers and backup software, were written for NetWare versions previous to NetWare 4. These applications used the NetWare bindery instead of eDirectory for network access and object manipulation.

The bindery is a flat database of objects such as Users, Groups, and Volumes known to a given server. The bindery is server-specific and server-centric.

Older NetWare client software (such as the NETX bindery shell) used a bindery login procedure in which a user logged in to a specific server only. Access to multiple servers required multiple logins using multiple user accounts.

eDirectory allows applications written for a bindery to function using bindery services. Bindery services allows you to set an eDirectory context or a number of contexts (up to 12) as an eDirectory server's virtual bindery. The context you set is called the server's bindery context.

Following are some important facts about bindery services:

- To use bindery services, you must set a bindery context for the eDirectory server.

- Not all objects map to bindery objects. Many objects, such as Alias objects, do not have a bindery equivalent.

- Most bindery applications have been upgraded to work with eDirectory. Check with your application vendor to get the newest version.

- Each eDirectory server with a bindery context must hold a master or read/write replica of the partition that includes the bindery context.

Synchronize Servers in the Replica Ring

When multiple servers hold replicas of the same partition, those servers are considered a replica ring. eDirectory automatically keeps those servers synchronized, so the object data is consistent on all replicas.

The following eDirectory processes keep servers in the replica ring synchronized.

- Replica synchronization

    For more information on replica synchronization, refer to .

- Schema synchronization

- Limber

- Backlink

    For more information on Backlink, refer to .

- Connection management

Access to Resources

eDirectory provides a basic level of network access security through default rights. You can provide additional access control by completing the tasks outlined below.

- Assigning rights

    Each time a user attempts to access a network resource, the system calculates the user's effective rights to that resource. To ensure that users

have the appropriate effective rights to resources, you can make explicit trustee assignments, grant security equivalences, and filter inherited rights.

To simplify the assignment of rights, you can create Group and Organizational Role objects, then assign users to the groups and roles.

◆ Adding login security

Login security is not provided by default. You can set up several optional login security measures, including login passwords, login location and time restrictions, limits on concurrent login sessions, intruder detection, and login disabling.

◆ Setting up role-based administration

You can set up administrators for specific object properties and grant them rights to only those properties. This allows you to create administrators with specific responsibilities that can be inheritable to subordinates of any given container object. A role-based administrator can have responsibilities over any specific properties, such as those that relate to employee information or passwords.

See "Administration Basics" in *ConsoleOne User Guide*.

You can also define roles in terms of the specific tasks that administrators can perform in role-based administration applications. See "Configuring Role-Based Administration" in *ConsoleOne User Guide*.

eDirectory Rights

When you create a tree, the default rights assignments give your network generalized access and security. Some of the default assignments are as follows:

◆ User Admin has the Supervisor right to the top of the tree, giving Admin complete control over the entire directory. Admin also has the Supervisor right to the NetWare Server object, giving complete control over any volumes on that server.

◆ [Public] has the Browse right to the top of the tree, giving all users the right to view any objects in the tree.

◆ Objects created through an upgrade process such as a NetWare migration, printing upgrade, or Windows NT user migration receive trustee assignments appropriate for most situations.

Trustee Assignments and Targets

The assignment of rights involves a trustee and a target object. The trustee represents the user or set of users that are receiving the authority. The target represents those network resources the users have authority over.

- If you make an Alias a trustee, the rights apply only to the object the alias represents. The Alias object can be an explicit target, however.

- A file or directory in the NetWare file system can also be a target, although file system rights are stored in the file system itself, not in eDirectory.

See "Administration Basics" in *ConsoleOne User Guide*.

The [Public] trustee is not an object. It is a specialized trustee that represents any network user, logged in or not, for rights assignment purposes.

eDirectory Rights Concepts

The following list of concepts helps you understand eDirectory rights.

Object (Entry) Rights

When you make a trustee assignment, you can grant object rights and property rights. Object rights apply to manipulation of the entire object, while property rights apply only to certain object properties. An object right is described as an entry right because it provides an entry in the eDirectory database.

A description of each object right follows.

- **Supervisor:** Includes all rights to the object and all of its properties.

- **Browse:** Lets the trustee see the object in the tree. It does not include the right to see an object's properties.

- **Create:** Applies only when the target object is a container. Create allows the trustee to create new objects below the container, and also includes the Browse right.

- **Delete:** Lets the trustee delete the target from the directory.

- **Rename:** Lets the trustee change the name of the target.

Property Rights

When you make a trustee assignment, you can grant object rights and property rights. Object rights apply to manipulation of the entire object, while property rights apply only to certain object properties.

ConsoleOne gives you two options for managing property rights:

 • You can manage all properties at once when the [All Attributes Rights] item is selected.

 • You can manage one or more individual properties when the specific property is selected.

The following are descriptions of each property right:

 • **Supervisor:** Gives the trustee complete power over the property.

 • **Compare:** Lets the trustee compare the value of a property to a given value. This right allows searching and returns only a true or false result. It does not allow the trustee to actually see the value of the property.

 • **Read:** Lets the trustee see the values of a property. It includes the Compare right.

 • **Write:** Lets the trustee create, change, and delete the values of a property.

 • **Add Self:** Lets the trustee add or remove itself as a property value. It only applies to properties with object names as values, such as membership lists or Access Control Lists (ACLs).

Effective Rights

Users can receive rights in a number of ways, such as explicit trustee assignments, inheritance, and security equivalence. Rights can also be limited by Inherited Rights Filters and changed or revoked by lower trustee assignments. The net result of all these actions–the rights a user can employ–are called *effective rights*.

A user's effective rights to an object are calculated each time the user attempts an action.

How eDirectory Calculates Effective Rights

Each time a user attempts to access a network resource, eDirectory calculates the user's effective rights to the target resource using the following process:

1. eDirectory lists the trustees whose rights are to be considered in the calculation. These include:

   ◆ The user who is attempting to access the target resource

   ◆ The objects that the user is security equivalent to

2. For each trustee in the list, eDirectory determines its effective rights as follows:

   a. eDirectory starts with the inheritable rights that the trustee has at the top of the tree.

      eDirectory checks the Object Trustees (ACL) property of the Tree object for entries that list the trustee. If any are found and they are inheritable, eDirectory uses the rights specified in those entries as the initial set of effective rights for the trustee.

   b. eDirectory moves down a level in the branch of the tree that contains the target resource.

   c. eDirectory removes any rights that are filtered at this level.

      eDirectory checks the ACL at this level for Inherited Rights Filters (IRFs) that match with the right types (object, all properties, or a specific property) of the trustee's effective rights. If any are found, eDirectory removes from the trustee's effective rights any rights that are blocked by those IRFs.

      For example, if the trustee's effective rights so far include an assignment of Write All Properties, but an IRF at this level blocks Write All Properties, the system removes Write All Properties from the trustee's effective rights.

   d. eDirectory adds any inheritable rights that are assigned at this level, overriding as needed.

      eDirectory checks the ACL at this level for entries that list the trustee. If any are found, and they are inheritable, eDirectory copies the rights from those entries to the trustee's effective rights, overriding as needed.

For example, if the trustee's effective rights so far include the Create and Delete object rights but no property rights, and if the ACL at this level contains both an assignment of zero object rights and an assignment of Write all properties for this trustee, then the system replaces the trustee's existing object rights (Create and Delete) with zero rights and adds the new all property rights.

    e. eDirectory repeats the filtering and adding steps (c and d above) at each level of the tree, including at the target resource.

    f. eDirectory adds any noninheritable rights assigned at the target resource, overriding as needed.

eDirectory uses the same process as in Step 2d above. The resulting set of rights constitutes the effective rights for this trustee.

3. eDirectory combines the effective rights of all the trustees in the list as follows:

    a. eDirectory includes every right held by any trustee in the list, and excludes only those rights that are missing from every trustee in the list. eDirectory does not mix right types. For example, it does not add rights for a specific property to rights for all properties or vice versa.

    b. eDirectory adds rights that are implied by any of the current effective rights.

The resulting set of rights constitutes the user's effective rights to the target resource.

Example

User DJones is attempting to access volume Acctg_Vol. See Figure 22.

Figure 22   Sample Trustee Rights
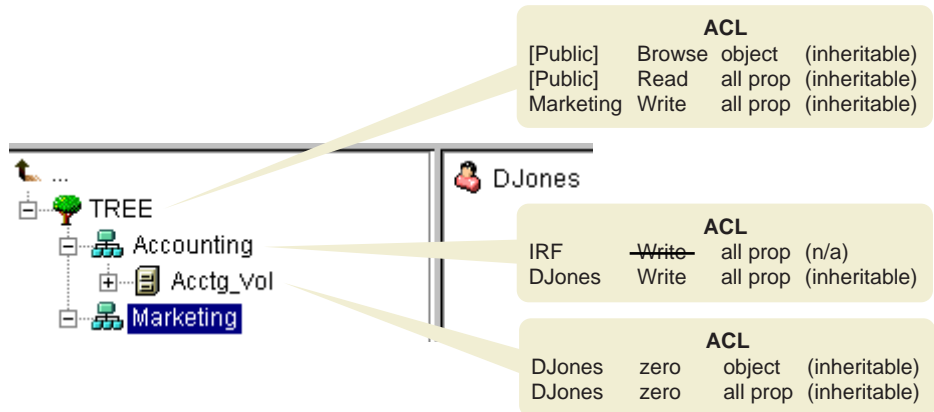
The following process shows how eDirectory calculates DJones' effective rights to Acctg_Vol:

1. The trustees whose rights are to be considered in the calculation are DJones, Marketing, Tree, and [Public].

   This assumes that DJones doesn't belong to any groups or roles and has not been explicitly assigned any security equivalences.

2. The effective rights for each trustee are as follows:

   ◆ DJones: Zero object, zero all properties

   The assignment of zero all property rights at Acctg_Vol overrides the assignment of Write all properties at Accounting.

   ◆ Marketing: Zero all properties

   The assignment of Write all properties at the top of the tree is filtered out by the IRF at Accounting.

   ◆ Tree: No rights

   No rights are assigned for Tree anywhere in the pertinent branch of the tree.

   ◆ [Public]: Browse object, Read all properties

   These rights are assigned at the root and aren't filtered or overridden anywhere in the pertinent branch of the tree.

3. Combining the rights from all these trustees results in the following:

   DJones: Browse object, Read all properties

4. Adding the Compare all properties right that is implied by the Read all properties right, DJones has the following final effective rights to Acctg_Vol:

DJones: Browse object, Read and Compare all properties

Blocking Effective Rights

Because of the way that effective rights are calculated, it is not always obvious how to block particular rights from being effective for specific users without resorting to an IRF (an IRF blocks rights for all users).

To block particular rights from being effective for a user without using an IRF, do either of the following:

- Ensure that neither the user nor any of the objects that the user is security equivalent to ever gets assigned those rights, either at the target resource or at any level above the target resource in the tree.

- If the user or any object that the user is security equivalent to does get assigned those rights, ensure that that object also has an assignment lower in the tree that omits those rights. Do this for every trustee (associated with the user) that has the unwanted rights.

Security Equivalence

Security equivalence means having the same rights as another object. When you make one object security equivalent to another object, the rights of the second object are added to the rights of the first object when the system calculates the first object's effective rights.

For example, suppose you make User object Joe security equivalent to the Admin object. After you create the security equivalence, Joe has the same rights to the tree and file system as Admin.

There are three types of security equivalence:

- Explicit: By assignment

- Automatic: By membership in a group or role

- Implied: Equivalent to all parent containers and the [Public] trustee

Security equivalence is only effective for one step. For example, if you make a third user security equivalent to Joe in the example above, that user does not receive Admin rights.

Security equivalence is recorded in eDirectory as values in the User object's Security Equal To property.

When you add a User object as an occupant to an Organizational Role object, that User automatically becomes security equivalent to the Organizational Role object. The same is true when a User becomes a member of a Group role object.

Access Control List (ACL)

The Access Control List (ACL) is also called the Object Trustees property. Whenever you make a trustee assignment, the trustee is added as a value to the Object Trustees (ACL) property of the target.

This property has strong implications for network security for the following reasons:

- ◆ Anyone who has the Supervisor or Write right to the Object Trustees (ACL) property of an object can determine who is a trustee of that object.

- ◆ Any users with the Add Self right to the Object Trustees (ACL) property of an object can change their own rights to that object. For example, they can grant themselves the Supervisor right.

For these reasons, be careful giving Add Self rights to all properties of a container object. That assignment makes it possible for the trustee to become Supervisor of that container, all objects in it, and all objects in containers beneath it.

Inherited Rights Filter (IRF)

The Inherited Rights Filter allows you to block rights from flowing down the eDirectory Tree. For more information on configuring this filter, see "Blocking Inheritance" in "Administering Rights in *ConsoleOne User Guide*.

Default Rights for a New Server

When you install a new Server object into a tree, the following trustee assignments are made:

Table 23   **Trustee Assignments**

Default Trustees	Default Rights
Admin (first eDirectory server in the tree)	Supervisor object right to the Tree object.
	Admin has the Supervisor object right to the NetWare Server object, which means that Admin also has the Supervisor right to the root directory of the file system of any volumes on the server.
[Public] (first eDirectory server in the tree)	Browse object right to the Tree object.
Tree	The Tree Read property right to the Host Server Name and Host Resource properties on all Volume objects.
	This gives all objects access to the physical volume name and physical server name.
Container objects	Read and File Scan rights to SYS: \PUBLIC. This allows User objects under the container to access NetWare utilities in \PUBLIC.
User objects	If home directories are automatically created for users, the users have the Supervisor right to those directories.

Delegated Administration

eDirectory lets you delegate administration of a branch of the tree, revoking your own management rights to that branch. One reason for this approach is that special security requirements require a different administrator with complete control over that branch.

To delegate administration:

1 Grant the Supervisor object right to a container.

2 Create an IRF on the container that filters the Supervisor and any other rights you want blocked.

IMPORTANT: If you delegate administration to a User object and that object is subsequently deleted, there are no objects with rights to manage that branch.

To delegate administration of specific eDirectory properties, such as Password Management, see "Granting Equivalence" in *ConsoleOne User Guide*.

To delegate the use of specific functions in role-based administration applications, see "Configuring Role-Based Administration" in *ConsoleOne User Guide*.

4 Managing Objects

Novell® eDirectory™ includes ConsoleOne™ 1.3, which allows you to manage the objects in your eDirectory tree. ConsoleOne 1.3 completely replaces NetWare® Administrator and NDS Manager™. To understand the new features and benefits of ConsoleOne 1.3, see "What's New in This Release?" and "Why Use ConsoleOne?" in *ConsoleOne User Guide*.

Managing eDirectory objects involves creating, modifying, and manipulating objects. For example, you might need to create user accounts and administer user rights. *ConsoleOne User Guide* provides detailed information on:

- Administration Basics: How to browse, create, edit, and organize objects
- Creating User Accounts: How to create an account to specify a user's login name and supply other information used by eDirectory.
- Administering Rights: How to assign rights, grant equivalence, block inheritance, and view effective rights.
- Configuring Role-Based Administration: How to define administrator's roles for specific administrative applications through the role-based services (RBS) objects.

General Object Tasks

This section contains steps for basic tasks you will use when managing your eDirectory tree, such as:

The *ConsoleOne User Guide* provides more information about completing other tasks, such as:

◆ Creating specific types of containers. For more information, refer to "Organizing Objects into Containers" in *ConsoleOne User Guide*.

◆ Extending objects with auxiliary class attributes

See "Extending an Object with the Properties of an Auxiliary Class" and "Extending Multiple Objects Simultaneously with the Properties of an Auxiliary Class" in *ConsoleOne User Guide*.

◆ Modifying an object's auxiliary properties

See "Modifying an Object's Auxiliary Properties" in *ConsoleOne User Guide*.

◆ Deleting auxiliary attributes from objects

See "Deleting Auxiliary Properties from an Object" and "Deleting Auxiliary Properties from Multiple Objects Simultaneously" in *ConsoleOne User Guide*.

Browsing the eDirectory Tree

In the left pane of ConsoleOne you'll see the "NDS" container, which holds the eDirectory trees that you are currently logged in to. You can cause additional eDirectory trees to appear in the "NDS" container by logging into those trees.

Once you are in an eDirectory tree or context and its objects are listed in the right pane, you can use the techniques described below to locate the specific objects you want to manage.

For more information about locating objects in the eDirectory tree, see "Browsing and Finding Objects" in *ConsoleOne User Guide*.

Creating an Object

1 In ConsoleOne, right-click the container that you want to create the object in > click New > Object.

2 Under Class, select the type of object > click OK.

3 If you get a warning that no snap-in is available to create the object, complete the appropriate action from Table 24, depending on your level of understanding of the object you are creating. If you don't see a warning, skip this step.

Table 24   Snap-in Availability Warning

Understanding Level	Action
Thorough: You understand this object type and how its properties are used.	Click Yes in the warning box. You will be allowed to set the object's mandatory properties using generic editors. After creating the object, you can set other properties using the generic Other property page.
Minimal: You understand what the object is, but not how its properties are used in any detail.	Click No in the warning box > quit this procedure. You will need to install a product that provides a ConsoleOne snap-in to create and manage this object type.

4 In Name, enter a name for the new object.

If it's an eDirectory object, be sure to follow proper naming conventions.

5 Specify any other information requested in the dialog box.

Click Help for more information. (If you are using generic editors, no information is available.)

6 Click OK.

For more information, see "Creating and Manipulating Objects" in *ConsoleOne User Guide*

Modifying an Object's Properties

1 In ConsoleOne, right-click the object > click Properties.

2 Edit the property pages you want.

Click Help for more information on specific properties.

3 Click OK.

Moving Objects

1 In the ConsoleOne right pane, Shift-click or Ctrl-click the objects to select them.

2 Right-click your selection > click Move.

3 Click the browse button next to the Destination field > select the container to move the objects to > click OK.

4 If you want to create an Alias in the old location for each object being moved, select Create an Alias for All Objects Being Moved.

This allows any operations that are dependent on the old location to continue uninterrupted until you can update those operations to reflect the new location.

5 Click OK.

Deleting Objects

1 In ConsoleOne, Shift-click or Ctrl-click the objects to select them.

You can't delete a container object unless you first delete all its contents.

2 Right-click your selection > click Delete.

3 In the confirmation dialog box, click Yes.

5 Managing the Schema

The schema of your Novell® eDirectory™ tree defines the classes of objects that the tree can contain, such as Users, Groups, and Printers. It specifies the attributes (properties) that comprise each object type, including those that are required when creating the object and those that are optional.

You might need to make changes to your schema as your organization's informational needs change. For example, if you never required a fax number on your User object before, but you need one now, you can create a new User class that has Fax Number as a mandatory attribute, then begin using the new User class to create User objects.

Schema Manager lets those with the Supervisor right to a tree customize the schema of that tree. Schema Manager is available from the Tools menu in ConsoleOne™.

You can use Schema Manager to:

- View a list of all classes and attributes in the schema.
- Extend the schema by adding a class or an attribute to the existing schema.
- Create a class by naming it and specifying applicable attributes, flags, and containers to which it can be added, and parent classes from which it can inherit attributes.
- Create an attribute by naming it and specifying its syntax and flags.
- Add an attribute to an existing class.
- Delete a class or an attribute that is not in use or that has become obsolete.
- Identify and resolve potential problems.

Extending the Schema

You can extend the schema of a tree by using ConsoleOne to create a new class or attribute. To extend the schema of your eDirectory tree, you need the Supervisor right to the entire tree.

You can extend the schema by:

- "Creating a Class" on page 148
- "Deleting a Class" on page 149
- "Creating an Attribute" on page 149
- "Adding an Optional Attribute to a Class" on page 150
- "Deleting an Attribute" on page 150

You can extend the schema for auxiliary attributes by:

- "Creating an Auxiliary Class" on page 151
- "Extending an Object with the Properties of an Auxiliary Class" on page 152
- "Extending Multiple Objects Simultaneously with the Properties of an Auxiliary Class" on page 153
- "Modifying an Object's Auxiliary Properties" on page 154
- "Deleting Auxiliary Properties from an Object" on page 155
- "Deleting Auxiliary Properties from Multiple Objects Simultaneously" on page 156

Creating a Class

You can add a class to your existing schema as your organizational needs change. The Create a Class Wizard in ConsoleOne helps you accomplish this task.

1 In ConsoleOne, click anywhere in the eDirectory tree whose schema you want to extend.

2 Click Tools > Schema Manager.

3 Click the Classes tab > Create.

4 Follow the instructions in the wizard to define the object class.

Help is available throughout the wizard.

See "Defining a Custom Object Class" in *ConsoleOne User Guide* for more information.

If you need to define custom properties to add to the object class, cancel the wizard and define the custom properties first. See "Creating an Attribute" on page 149 for more information.

Deleting a Class

You can delete unused classes that aren't part of the base schema of your eDirectory tree. ConsoleOne only prevents you from deleting classes that are currently being used in locally replicated partitions.

You might also want to consider deleting a class from the schema in the following instances:

 ◆ After merging two trees and resolving class differences

 ◆ Any time a class has become obsolete

To delete a class:

1 In ConsoleOne, click anywhere in the eDirectory tree whose schema you want to modify.

2 Click Tools > Schema Manager.

3 Click the Classes tab > select the class > click Delete > click Yes.

See "Deleting a Class from the Schema" in *ConsoleOne User Guide* for more information.

Creating an Attribute

You can define your own custom types of properties and add them as optional properties to existing object classes. You can't, however, add mandatory properties to existing classes. The Create an Attribute Wizard in ConsoleOne helps you accomplish this task.

1 In ConsoleOne, click anywhere in the eDirectory tree whose schema you want to extend.

2 Click Tools > Schema Manager.

3 Click the Attributes tab > Create.

4 Follow the instructions in the wizard to define the new property.

Help is available throughout the wizard.

See "Defining a Custom Property" in *ConsoleOne User Guide* for more information.

Adding an Optional Attribute to a Class

You can add optional attributes to existing classes. This might be necessary if:

◆ Your organization's informational needs change

◆ You are preparing to merge trees

Mandatory attributes can only be defined while creating a class.

1 In ConsoleOne, click anywhere in the eDirectory tree whose schema you want to extend.

2 Click Tools > Schema Manager.

3 Click the Classes tab > select the class you want to modify > click Add.

4 In the list on the left, double-click the properties you want to add.

If you add a property by mistake, double-click it in the list on the right.

5 Click OK.

Objects you create of this class will now have the properties you added. To set values for the added properties, use the generic Other property page of the object.

See "Adding Optional Properties to a Class" in *ConsoleOne User Guide* for more information.

Deleting an Attribute

You can delete unused attributes that aren't part of the base schema of your eDirectory tree.

You might also want to delete an attribute from the schema in the following instances:

◆ After merging two trees and resolving attribute differences.

◆ Any time an attribute has become obsolete.

To delete an attribute:

1 In ConsoleOne, click anywhere in the eDirectory tree whose schema you want to modify.

2 Click Tools > Schema Manager.

3 Click the Attributes tab > select the property > click Delete > click Yes.

See "Deleting a Property from the Schema" in *ConsoleOne User Guide* for more information.

Creating an Auxiliary Class

An auxiliary class is a set of properties (attributes) that is added to particular eDirectory object instances rather than to an entire class of objects. For example, an e-mail application could extend the schema of your eDirectory tree to include an E-Mail Properties auxiliary class and then extend individual objects with those properties as needed.

With Schema Manager, you can define your own auxiliary classes. You can then extend individual objects with the properties defined in your auxiliary classes.

1 In ConsoleOne, click anywhere in the eDirectory tree whose schema you want to extend.

2 Click Tools > Schema Manager.

3 Click the Classes tab > Create.

4 Follow the instructions in the wizard to define the auxiliary class.

Make sure to select Auxiliary Class when setting the class flags. If you need to define custom properties to add to the auxiliary class, cancel the wizard and define the custom properties first. See "Creating a Class" on page 148 for more information.

For more detailed information on defining and using auxiliary classes, see "Defining an Auxiliary Class" in *ConsoleOne User Guide*.

Extending an Object with the Properties of an Auxiliary Class

1 In the main ConsoleOne window, right-click the object > click Extensions of This Object.

2 Depending on whether the auxiliary class that you want to use is already listed under Current Auxiliary Class Extensions, complete the appropriate action:

Auxiliary Class Is Already Listed?	Action
Yes	Quit this procedure. See "Modifying an Object's Auxiliary Properties" on page 154 instead.
No	Click Add Extension > select the auxiliary class > click OK.

3 If a message appears stating that generic editors will be used, click OK.

4 On the screen that appears, set the property values you want. Depending on which screen you're using, note the following:

Screen	Notes
Extensions tab (Properties dialog box)	◆ Both mandatory and optional properties of the auxiliary class might be listed. ◆ Click Help for more information on specific properties.
New dialog box	◆ Only mandatory properties of the auxiliary class are listed. ◆ You must know the syntax of a property to set it correctly. For more information, see *NDS 8 Documentation* > Understanding Schema Manager (http://www.novell.com/documentation/lg/nds8/usnds/schm_enu/data/hnpkthb2.html). ◆ After setting the mandatory properties, you can set optional properties as explained in "Modifying an Object's Auxiliary Properties" on page 154.

5 Click OK.

See "Extending an Object with the Properties of an Auxiliary Class" in *ConsoleOne User Guide* for more information.

Extending Multiple Objects Simultaneously with the Properties of an Auxiliary Class

1 In the ConsoleOne right pane, Shift-click or Ctrl-click the objects to select them.

The objects don't need to be the same type.

2 Right-click your selection > click Extensions of Multiple Objects.

3 Depending on whether the auxiliary class that you want to use is already listed under Current Auxiliary Class Extensions, complete the appropriate action in Table 25.

Only those extensions that are common to all the selected objects are listed. Those that are specific to individual objects aren't listed.

Table 25   Action to Take if Auxiliary Class is Listed or Not Listed

Auxiliary Class Is Already Listed?	Action
Yes	Quit this procedure. See "Modifying an Object's Auxiliary Properties" on page 154 instead. You'll have to modify the objects one at a time.
No	Click Add Extension > select the auxiliary class > click OK.

4 If a message appears stating that generic editors will be used, click OK.

5 On the screen that appears, set the property values you want.

> **IMPORTANT:** Each property value you set will be applied to each selected object. If the property already exists in the object and is single-valued, the existing value will be replaced. If the property already exists and is multivalued, the new values will be added to the existing values.

Depending on which screen you're using, also note the following:

Screen	Notes
Extensions tab	◆ Both mandatory and optional properties of the auxiliary class might be listed.
	◆ Click Help for more information on specific properties.
New dialog box	◆ Both mandatory and optional properties of the auxiliary class might be listed.
	◆ Click Help for more information on specific properties.

6 Click OK.

See "Extending Multiple Objects Simultaneously with the Properties of an Auxiliary Class" in *ConsoleOne User Guide* for more information.

Modifying an Object's Auxiliary Properties

1 In the main ConsoleOne window, right-click the object > click Properties.

2 Click the Extensions tab > select the property page that's named after the auxiliary class.

If the auxiliary class isn't listed or if there's no Extensions tab, use the generic Other page.

3 On the screen that appears, set the property values you want. Depending on which screen you're using, note the following:

Screen	Notes
Extensions tab	◆ Both mandatory and optional properties of the auxiliary class might be listed. ◆ Click Help for more information on specific properties.
Other tab	◆ Only the properties of the auxiliary class that have already been set are listed. Click Add to set additional properties. ◆ You must know the syntax of a property to set it correctly. For more information, see *NDS 8 Documentation* > Understanding Schema Manager (http://www.novell.com/documentation/lg/nds8/usnds/schm_enu/data/hnpkthb2.html).

4 Click OK.

See "Modifying an Object's Auxiliary Properties" in *ConsoleOne User Guide* for more information.

Deleting Auxiliary Properties from an Object

1 In the main ConsoleOne window, right-click the object > click Extensions of This Object.

2 In the list of current auxiliary class extensions, select the auxiliary class whose properties you want to delete.

3 Click Remove Extension > Yes.

This deletes all the properties added by the auxiliary class except for any that the object already had innately.

See "Deleting Auxiliary Properties from an Object" in *ConsoleOne User Guide* for more information.

Deleting Auxiliary Properties from Multiple Objects Simultaneously

1 In the ConsoleOne right pane, Shift-click or Ctrl-click the objects to select them.

The objects don't need to be the same type.

2 Right-click your selection > click Extensions of Multiple Objects.

3 Depending on whether the auxiliary class whose properties you want to delete is listed under Current Auxiliary Class Extensions, complete the appropriate action from Table 26

Only those extensions that are common to all the selected objects are listed. Those that are specific to individual objects aren't listed.

Table 26   **Action to Take if Auxiliary Class is Listed or Not Listed**

Auxiliary Class Is Listed?	Action
Yes	Select the auxiliary class > click Remove Extension > click Yes. This deletes all the properties added by the auxiliary class except for any that the object already had innately.
No	Cancel the dialog box. You'll have to delete the auxiliary class from each object one at a time. For more information, see "Deleting Auxiliary Properties from an Object" on page 155.

See "Deleting Auxiliary Properties from Multiple Objects Simultaneously" in *ConsoleOne User Guide* for more information.

Viewing the Schema

You can view the schema to evaluate how well the schema meets your organization's informational needs. The larger and more complex your organization, the more likely it is that you need to customize the schema, but even small organizations might have unique tracking needs. Viewing the schema can help you determine what, if any, extensions you need to make to the base schema.

For information about the viewing and printing functionality of NDS Manager in previous versions, see *NDS eDirectory Administration Guide* (http://www.novell.com/documentation/lg/ndsse/ndsseenu/data/a2iiikq.html).

Viewing the Current Schema

1 In ConsoleOne, click anywhere in the eDirectory tree whose schema you want to view.

2 Click Tools > Schema Manager.

A list of the available classes and properties appears. Double-click a class or property to see information about it.

Extending the Schema on Linux or Solaris Systems

The following sections provide information about extending the schema on Linux* and Solaris* systems:

Using the ndssch Utility to Extend the Schema on Linux or Solaris

You can use ndssch, the eDirectory schema extension utility, to extend the schema on Linux or Solaris systems. The attributes and classes that you specify in the schema file (.SCH) will be used to modify the schema of the tree. The association between the attributes and classes are created as specified in the .SCH file.

To extend the schema:

1 Use the following syntax:

```
ndssch [-h <hostname>[:<port>]] [-t tree_name] admin-FDN
schemafile...

ndssch [-h <hostname>[:<port>]] [-t tree_name] [-d]
admin_FDN schemafile [schema_description]...
```

Table 27   **ndssch Parameters**

ndssch Parameters	Description
-h *hostname*	The name or IP address of the server on which the schema is to be extended. The schema of the tree that the specified server belongs to will be extended. This is an optional parameter if the tree is located on the host whose schema is to be extended, otherwise it is a mandatory parameter.
port	Specifies the server port.
-t *tree_name*	Name of the tree on which the schema is to be extended. This is an optional parameter. The default tree name is the one specified in the /etc/nds.conf file. For more information, see "Configuration Parameters" on page 49.
admin-FDN	Name with the full context of the user with eDirectory administrator rights to the Tree.
schemafile	Filename that contains information about the schema to be extended.
-d, schema_description	When this option is used, every schema file must be followed by a description of the schema file.

Extending the RFC 2307 Schema

The attributes and object classes defined in RFC 2307 (http://www.isi.edu/in-notes/rfc2307.txt) are user or group related and NIS related. The user or group related definitions are compiled into the /usr/lib/nds-modules/schema/rfc2307-usergroup.sch file. The NIS related definitions are compiled into the /usr/lib/nds-modules/schema/rfc2307-nis.sch file. The corresponding files in the LDIF format are also provided (/usr/lib/nds-modules/schema/rfc2307-usergroup.ldif and /usr/lib/nds-modules/schema/rfc2307-nis.ldif respectively).

You can extend the RFC 2307 schema using the ndssch utility or the ldapmodify tool.

To extend the schema using the ndssch utility:

1 Enter the following command:

```
ndssch -t /usr/lib/nds-modules/schema/rfc2307-
usergroup.sch
```

or

```
ndssch -t /usr/lib/nds-modules/schema/rfc2307-
nis.sch
```

Table 28   ndssch Parameters

ndssch Parameter	Description
-t	The name of the tree on which the schema is to be extended. This is an optional parameter. If this parameter is not specified, the tree name is taken from the /etc/nds.conf file.

To extend the schema using the ldapmodify utility:

1 Enter the following command:

```
ldapmodify -h -D -w -f /usr/lib/nds-modules/
schema/rfc2307-usergroup.ldif
```

or

```
ldapmodify -h -D -w -f /usr/lib/nds-modules/
schema/rfc2307-nis.ldif
```

Table 29   ldapmodify Parameters

ldapmodify Parameters	Description
-h *ldaphost*	Specify an alternate host on which the LDAP server is running.
-D *binddn*	Use binddn to bind to the X.500 directory. *binddn* should be a string-represented DN as defined in RFC 1779.
-w *passwd*	Use *passwd* as the password for simple authentication.

ldapmodify Parameters	Description
-f file	Read the entry modification information from file instead of from standard input.

6 Managing Partitions and Replicas

Partitions are logical divisions of the Novell® eDirectory™ database that form a distinct unit of data in the eDirectory tree that administrators use to store and replicate eDirectory information. Each partition consists of a container object, all objects contained in it, and the information about those objects. Partitions do not include any information about the file system or the directories and files contained there.

Instead of storing a copy of the entire eDirectory database on each server, you can make a copy of the eDirectory partition and store it on many servers across the network. Each copy of the partition is known as a replica. You can create any number of replicas for each eDirectory partition and store them on any server. The types of replicas include master, read/write, read only, subordinate references, filtered read/write, and filtered read only.

Table 30 describes the replica types.

Table 30 **Replica Types**

Replicas	Description
Master, read/write, and read only replicas	Contain all objects and attributes for a particular partition.
Subordinate references	Used for tree connectivity.

Replicas	Description
Filtered replicas	Contain a subset of information from the entire partition, consisting of only the desired classes and attributes. Desired classes and attributes are defined by the server's replication filter, which is used to identify which classes and attributes are allowed to pass during inbound synchronization and local changes.
	Filtered replicas allow administrators to create sparse and fractional replicas.
	◆ Sparse replicas: Contain only the object classes that you specify.
	◆ Fractional replicas: Contain only the attributes you specify.
	The functionality of filtered replicas enables fast response when the data stored in eDirectory is procured by applications. Filtered replicas also allow more replicas to be stored on a single server.
Read/write filtered replicas	Allow local modifications to classes and attributes that are a subset of the server's replication filter. However, these replicas can only create objects if all mandatory attributes for the class are within the replication filter.
Read only filtered replicas	These replicas do not allow local modifications.

This section describes how to manage partitions and replicas.

Creating a Partition

When you create partitions, you make logical divisions of your tree. These divisions can be replicated and distributed among different eDirectory servers in your network.

When you create a new partition, you split the parent partition and end up with two partitions. The new partition becomes a child partition. See .

Figure 23   Before and After a Partition Split
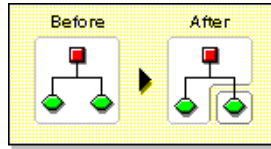
For example, if you choose an Organizational Unit and create it as a new partition, you split the Organizational Unit and all of its subordinate objects from its parent partition.

The Organizational Unit you choose becomes the root of a new partition. The replicas of the new partition exist on the same servers as the replicas of the parent, and objects in the new partition belong to the new partition's root object.

Creating a partition might take some time, since all of the replicas need to be synchronized with the new partition information. If you attempt another partition operation while a partition is still being created, you receive a message telling you that the partition is busy.

You can look at the replica list for the new partition and know that the operation is complete when all replicas in the list are in an On state. You must manually refresh the view periodically because the states are not automatically refreshed.

You can create a partition only from the Tree view.

For more information, see "Splitting a Partition (Creating a Child Partition)" in *ConsoleOne User Guide*.

Merging a Partition

When you merge a partition with its parent partition, the chosen partition and its replicas combine with the parent partition. You do not delete partitions–you only merge and create partitions to define how the directory tree is split into logical divisions. See Figure 24 on page 163.

Figure 24   Before and After a Partition Merge
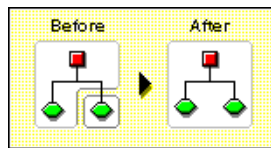
There are several reasons you might want to merge a partition with its parent:

- The directory information in the two partitions is closely related.

- You want to delete a subordinate partition, but you don't want to delete the objects in it.

- You're going to delete the objects in the partition.

- You want to delete all replicas of the partition. (Merging a partition with its parent is the only way to delete the partition's master replica.)

- After moving a container (which must be a partition root with no subordinate partitions), you don't want the container to be a partition anymore.

- You experience changes in your company organization, so you want to redesign your directory tree by changing the partition structure.

Consider keeping partitions separate if the partitions are large (contain hundreds of objects), because large partitions slow down network response time.

The root-most partition in the tree cannot be merged because it is the top partition and has no parent partition to merge with.

The partition is merged when the process is completed on the servers. The operation could take some time to complete depending on partition sizes, network traffic, server configuration, and so on.

IMPORTANT: Before merging a partition, check the partition synchronization of both partitions and fix any errors before proceeding. By fixing the errors, you can isolate problems in the directory and avoid propagating the errors or creating new ones.

Make sure all servers that have replicas (including subordinate references) of the partition you want to merge are up before attempting to merge a partition. If a server is down, eDirectory won't be able to read the server's replicas and won't be able to complete the operation.

If you receive errors in the process of merging a partition, resolve the errors as they appear. Don't try to fix the error by continuing to perform operations–doing so only results in more errors.

For more information, see "Merging a Child Partition with Its Parent Partition" in *ConsoleOne User Guide*.

Moving Partitions

Moving a partition lets you move a subtree in your directory tree. You can move a partition root object (which is a container object) only if it has no subordinate partitions. See Figure 25.

Figure 25   Before and After a Partition Move
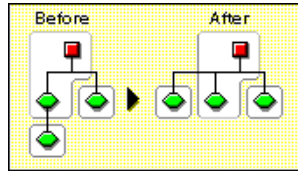
When you move a partition, you must follow eDirectory containment rules. For example, you cannot move an Organizational Unit directly under the root of the current tree, because the root's containment rules allow Locality, Country, or Organization, not Organizational Unit.

When you move a partition, eDirectory changes all references to the partition root object. Although the object's common name remains unchanged, the complete name of the container (and of all its subordinates) changes.

When you move a partition, you should probably choose the option to create an Alias object in place of the container you're moving. Doing so allows users to continue to log in to the network and find objects in their original directory location.

The Alias object that is created has the same common name as the moved container and references the new complete name of the moved container.

IMPORTANT: If you move a partition and do not create an Alias object in place of the moved partition, users who are unaware of the partition's new location cannot easily find that partition's objects in the directory tree, since they look for them in their original directory location.

This might also cause client workstations to fail at login if the workstation NAME CONTEXT parameter is set to the original location of the container in the directory tree.

Because the context of an object changes when you move it, users whose name context references the moved object need to update their NAME CONTEXT parameter so that it references the object's new name.

To automatically update users' NAME CONTEXT after moving a container object, use the NCUPDATE utility.

After moving the partition, if you don't want the partition to remain a partition, merge it with its parent partition.

Make sure your directory tree is synchronizing correctly before you move a partition. If you have any errors in synchronization in either the partition you want to move or the destination partition, do not perform a move partition operation. First, fix the synchronization errors.

For more information, see "Moving a Partition in *ConsoleOne User Guide.*

Aborting Create or Merge Partition Operations

You can abort a Create or Merge partition operation if the operation has not yet progressed past the stage at which the change is committed. Use this feature to back out of an operation, or if your eDirectory network returns eDirectory errors or fails to synchronize following a partition operation.

If replicas in your directory tree experience synchronization errors, an abort operation might not always solve the problem. However, you can use this feature as an initial troubleshooting option.

If a partition operation cannot be completed because a server is down (or otherwise unavailable), either make the server visible to the network so the operation can complete or attempt to abort the operation. If eDirectory cannot synchronize because the database is corrupted, you should abort any partition operation in progress.

Partition operations can take considerable time to fully synchronize across the network, depending on the number of replicas involved, the visibility of servers involved, and the existing wire traffic.

If you get an error that says a partition is busy, it doesn't mean that you should abort the operation. You can usually expect partition operations to complete within 24 hours depending on the size of the partition, connectivity issues, etc. If a particular operation fails to complete within this time frame, you should then attempt to abort the operation in progress.

Adding, Deleting, and Changing the Type of Replicas

Before you add or delete a replica, or change replica type, carefully plan target replica locations. See "Guidelines for Replicating Your Tree" on page 69.

Adding a Replica

Add a replica to a server to provide your directory with

- Fault tolerance

- Faster access to data

- Faster access across a WAN link

- Access to objects in a set context (using bindery services)

For instructions on adding a replica, see "Adding a Replica" in *ConsoleOne User Guide*.

Deleting a Replica

Deleting a replica removes the replica of the partition from a server.

If you want to remove a server from the directory tree, you can delete replicas from the server before removing it. Removing the replicas reduces the chance of having problems when removing the server.

You can also reduce synchronization traffic on the network by removing replicas. Keep in mind that you probably don't want more than six replicas of any partition.

You cannot delete a master replica or a subordinate reference.

If the replica you want to delete is a master, you have two options:

- Go to a server with another replica of the partition and make it the new master replica.

  This automatically changes the original master replica to a read/write replica, which you can then delete.

- Merge the partition with its parent partition.

  This merges the replicas of the partition with those of its parent and removes them from the servers they reside on. Merging removes partition boundaries, but not the objects. The objects continue to exist on each server which held a replica of the "joined" partition.

When you delete replicas, keep the following guidelines in mind:

- For fault tolerance, you should maintain at least three replicas of each partition on different servers.

- Deleting a replica deletes a copy of part of the directory database on the targeted server.

  The database can still be accessed on other servers in the network, and the server that the replica was on still functions in eDirectory.

  You cannot delete or manage subordinate reference replicas. They are created automatically on a server by eDirectory when the server contains a replica of a partition but not of that partition's child.

For more information on deleting a replica, see "Deleting a Replica" in *ConsoleOne User Guide*.

Changing a Replica Type

Change a replica type to control access to the replica information. For example, you might want to change an existing read/write replica to a read-only replica to prevent users from writing to the replica and modifying directory data.

You can change the type of a read/write or a read-only replica. You cannot change the type of a master replica, but a read/write or read-only can be changed to a master, then the original master is changed automatically to a read/write replica.

Most replicas should be read/write. Read/write replicas can be written to by client operations. They send out information for synchronization when a change is made. Read-only replicas cannot be written to by client operations. However, they are updated when the replicas synchronize.

You cannot change the replica type of a subordinate reference. To place a replica of a partition on a server which currently has a subordinate reference requires an Add replica operation. A subordinate reference replica is not a complete copy of a partition. The placement and management of subordinate reference replicas is handled by eDirectory. They are created automatically on a server by eDirectory when the server contains a replica of a partition but not of that partition's child.

You can't use this procedure to change the type of the master replica. To specify a new master replica, change the type of an existing read/write or read-only replica to master, and the original master replica is automatically changed to read/write.

See "Modifying a Replica" in *ConsoleOne User Guide*.

Setting Up and Managing Filtered Replicas

Filtered replicas maintain a filtered subset of information from an eDirectory partition (objects or object classes along with a filtered set of attributes and values for those objects).

Administrators will generally use the filtered replica capability to create an eDirectory server that holds a set of filtered replicas that contain only specific objects and attributes that they want synchronized.

To do this, ConsoleOne provides a snap-in with the ability to create a filtered replica partition scope and a filter. A scope is simply the set of partitions on which you want replicas placed on a server, while a replication filter contains the set of eDirectory classes and attributes you want to host on a server's set of filtered replicas. The result is an eDirectory server that can house a well-defined data set from many partitions in the tree.

The descriptions of the server's partition scope and replication filters are stored in eDirectory, and they can be managed through the Server object in ConsoleOne.

- "Using the Filtered Replica Configuration Wizard" on page 169
- "Defining a Partition Scope" on page 170
- "Setting Up a Server Filter" on page 171

Using the Filtered Replica Configuration Wizard

The Filtered Replica Configuration Wizard guides you step-by-step through the setup of a server's replication filter and partition scope.

1 In ConsoleOne, click Wizards > Filtered Replica Configuration.

2 Select the Server object that will host the filtered replicas > click Next.

3 To define the replication filter for this server, click Define the Filter Set.

The replication filter contains the set of eDirectory classes and attributes you want to host on this server's set of filtered replicas. For more information on defining a filter set, see "Setting Up a Server Filter" on page 171.

4 Click Edit Filter > add the classes and filters you want > click OK.

5 Click Apply > OK > Next.

6 To define the partition scope for this server, click Define the Partition Scope.

The partition scope is the set of partitions on which you would like replicas placed on this server. For more information on partition scopes, see "Defining a Partition Scope" on page 170.

7 Select a replica from the list > click Change Replica Type.

8 Click Filtered Read/Write or Filtered Read Only > OK.

9 Click Apply > OK > Next.

10 Once the replicas have been configured, click Finish.

Defining a Partition Scope

A partition scope is the set of partitions of which you want replicas placed on a server. The Replicas-Partition Scope window in ConsoleOne provides an expandable view of the hierarchy of partitions in the eDirectory tree. You can select individual partitions, a set of partitions of a given branch, or all of the partitions in the tree. You can then select the type of replicas of these partitions you want added to the server.

A server can hold both full replicas and filtered replicas. For more information on filtered replicas, see "Filtered Replicas" on page 129.

Viewing Replicas Present on an eDirectory Server

1 In ConsoleOne, right-click an eDirectory Server object.

2 Click Properties > Replicas-Partition Scope tab.

3 Select a partition.

4 View the list of replicas on this server.

Adding a Filtered Replica to an eDirectory Server

1 In ConsoleOne, right-click an eDirectory Server object.

2 Click Properties > Replicas-Partition Scope tab.

3 Select a replica from the list > click Change Replica Type.

4 Click Filtered Read/Write or Filtered Read Only.

5 Click OK.

6 Click Apply > OK.

Changing a Full Replica into a Filtered Replica

1 In ConsoleOne, right-click an eDirectory Server object.

2 Click Properties > Replicas-Partition Scope tab.

3 Select a replica from the list > click Change Replica Type.

4 Click Filtered Read/Write or Filtered Read Only.

5 Click OK > Apply > OK.

Setting Up a Server Filter

A server replication filter contains the set of eDirectory classes and attributes you want to host on a server's set of filtered replicas. You can set up a filter from any Server object. For filtered replicas, you can only have one filter per server. This means that any filter defined for an eDirectory server applies to all filtered replicas on that server. The filter, however, does not apply to full replicas.

A server's filter may be modified if required, but the operation generates a re-synchronization of the replica and may thus be time-consuming. Careful planning of the server's function is recommended.

1 In ConsoleOne, right-click an eDirectory Server object.

2 Click Properties > the Replicas-Filter tab > Edit Filter

3 Add the classes and filters you want > click OK.

4 Click Apply > OK.

Maintaining Partitions and Replicas

This section contains the following information:

- "Viewing the Partitions on a Server" on page 172
- "Viewing a Partition's Replicas" on page 172
- "Viewing Information about a Partition" on page 173
- "Viewing Partition Hierarchy" on page 173
- "Viewing Information about a Replica" on page 173

Viewing the Partitions on a Server

To view which partitions are allocated to a server:

1 In ConsoleOne, select a Server object.

2 Click View > Partition and Replica View.

This is a routine operation that is safe under all circumstances.

You might want to view the partitions stored on a server if you are planning to remove a Server object from the directory tree. In this case, you can view the replicas you need to remove before removing the object.

See "Viewing Replication Information" in *ConsoleOne User Guide.*

Viewing a Partition's Replicas

This operation lets you identify:

- Which servers the partition's replicas reside on
- Which server hosts the master replica of the partition
- Which servers have read/write, read-only, and subordinate reference replicas of the partition
- The state of each of the partition's replicas

See "Viewing Replication Information" in *ConsoleOne User Guide*.

Viewing Information about a Partition

This is a safe operation that you can perform under all circumstances.

The most significant reason to view information about a partition is to see its synchronization information. (You can gather most of the information about a partition without leaving the main view you're working from.)

For more information, see "Viewing Information about a Partition" in *ConsoleOne User Guide*.

Viewing Partition Hierarchy

You can easily view the partition hierarchy in ConsoleOne. You can expand container objects to view which partitions are parent, and which are child partitions.

Each container representing the root of a partition is marked with the following icon: .

Viewing Information about a Replica

This is a safe operation that you can perform under all circumstances.

The most significant reason to view information about a replica is to see its synchronization information. (You can gather most of the information about a replica without leaving the main view you're working from.)

You can also get additional help on the specific synchronization error by clicking the blue question mark button at the end of the error number line.

See "Viewing Replication Information" in *ConsoleOne User Guide*.

7 Novell eDirectory Management Utilities

This section contains information on the following Novell® eDirectory™ utilities:

- "Novell Import Conversion Export Utility" on page 175
- "Novell iMonitor" on page 208
- "Index Manager" on page 231
- "Predicate Data" on page 236

Novell Import Conversion Export Utility

The Novell Import Conversion Export utility lets you:

- Import data from LDIF files to the LDAP directory
- Export data from the LDAP directory to an LDIF file
- Migrate data between LDAP servers

The Novell Import Conversion Export utility manages a collection of handlers that read or write data in a variety of formats. Source handlers read data, while destination handlers write data. A single executable module can be both a source and a destination handler. The engine receives data from a source handler, processes the data, then passes the data to a destination handler.

For example, if you want to import LDIF data into an LDAP directory, the Novell Import Conversion Export engine uses an LDIF source handler to read an LDIF file and an LDAP destination handler to send the data to the LDAP directory server. See "Troubleshooting LDIF Files" on page 476 for more information on LDIF file syntax, structure, and debugging.

You can run the Novell Import Conversion Export client utility from the command line or from a snap-in to ConsoleOne™. The comma-delimited data handler, however, is a command line utility only.

You can use the Novell Import Conversion Export utility in either of the following ways:

- "Using the Novell eDirectory Import/Export Wizard" on page 176
- "Using the Command Line Interface" on page 180

The wizard and the command line interface both give you access to the Novell Import Conversion Export engine, but the command line interface gives you greater options for combining source and destination handlers.

The Novell Import Conversion Export utility replaces both the BULKLOAD and ZONEIMPORT utilities included with previous versions of NDS and eDirectory.

Using the Novell eDirectory Import/Export Wizard

The Novell eDirectory Import/Export Wizard is a ConsoleOne snap-in designed to help you:

- Import data from LDIF files to the LDAP directory
- Export data from the LDAP directory to an LDIF file
- Perform a LDAP server-to-LDAP server data migration

Performing an LDIF Import

1 In ConsoleOne, select Wizard > NDS Import/Export.

2 Click Import LDIF File > Next.

3 Enter the name of the LDIF file containing the data you want to import > click Next.

Click Advanced to set other LDIF source handler options. Click Help on the Advanced dialog box for more information on the available options.

4 Select the LDAP server where the data will be imported.

You can store information for several servers and name each set of information. Click New to add a new server.

5 Add the information from Table 31.

Table 31   LDAP Server Options

Option	Description
Server DNS name/IP address	Enter the DNS name or IP address of the destination LDAP server.
Port	Enter the integer port number of the destination LDAP server.
DER file containing server key used for SSL communication	Enter the name of the DER file containing a server key used for SSL authentication.
Login method	Click Authenticated Login or Anonymous for the entry specified in the User DN field.
User DN	Enter the distinguished name of the entry that should be used when binding to the server-specified bind operation.
Password	Enter the password attribute of the entry specified in the User DN field.

6 Click Next > Finish to begin the LDIF import.

Performing an LDIF Export

1 In ConsoleOne, select Wizard > NDS Import/Export.

2 Click Export LDIF File > Next.

3 Select the LDAP server holding the entries you want to export.

You can store information for several servers and name each set of information. Click New to add a new server. Click Advanced to set additional options for the LDAP source handler. Click Help on the Advanced dialog box for more information on the available options.

4 Add the information from .

Table 32   LDAP Server Options

Option	Description
Server DNS name/IP address	Enter the DNS name or IP address of the source LDAP server.
Port	Enter the integer port number of the source LDAP server.
DER file containing server key used for SSL communication	Enter the name of the DER file containing a server key used for SSL authentication.
Login method	Click Authenticated Login or Anonymous for the entry specified in the User DN field.
User DN	Enter the distinguished name of the entry that should be used when binding to the server-specified bind operation.
Password	Enter the password attribute of the entry specified in the User DN field.

5 Click Next.

6 Specify the search criteria for the entries you want to export.

Table 33   Search Criteria Options

Option	Description
Base DN	Enter the base distinguished name for the search request. If this field is left empty, the base DN defaults to "" (empty string).
Scope	Specifies the scope of the search request.
Filter	Enter an RFC 1558 compliant search filter. The default is objectclass=*.
Attributes	Specify the attributes you want returned for each search entry.

7 Click Next.

8 Enter the name of the LDIF file that will store the export information > click Next.

9 Click Finish to begin the LDIF export.

Performing Data Migration between LDAP Servers

1 In ConsoleOne, select Wizard > NDS Import/Export.

2 Click Migrate Data between LDAP Servers > Next.

3 Select the LDAP server holding the entries you want to migrate.

You can store information for several servers and name each set of information. Click New to add a new server. Click Advanced to set additional options for the LDAP source handler. Click Help on the Advanced dialog box for more information on the available options.

4 Add the information from Table 34.

Table 34 LDAP Server Options

Option	Description
Server DNS name/IP address	Enter the DNS name or IP address of the source LDAP server.
Port	Enter the integer port number of the source LDAP server.
DER file containing server key used for SSL communication	Enter the name of the DER file containing a server key used for SSL authentication.
Login method	Click Authenticated Login or Anonymous for the entry specified in the User DN field.
User DN	Enter the distinguished name of the entry that should be used when binding to the server-specified bind operation.
Password	Enter the password attribute of the entry specified in the User DN field.

5 Click Next.

6 Specify the following search criteria for the entries you want to migrate:

Table 35 **Search Criteria Options**

Option	Description
Base DN	Enter the base distinguished name for the search request. If this field is left empty, the base DN defaults to "" (empty string).
Scope	Specifies the scope of the search request.
Filter	Enter an RFC 2254 compliant search filter. The default is objectclass=*.
Attributes	Specifies the attributes you want returned for each search entry.

7 Click Next.

8 Select the LDAP server where the data will be migrated.

9 Click Next > Finish.

NOTE: Ensure that the schema is consistent across LDAP Services.

Using the Command Line Interface

You can use the command line version of the Novell Import Conversion Export utility to perform the following:

- LDIF imports

- LDIF exports

- Comma-delimited data imports

- Comma-delimited data exports

- Data migration between LDAP servers

The Novell Import Conversion Export utility is installed as part of ConsoleOne. Both a Win32* version (ICE.EXE) and a NetWare® version (ICE.NLM) are included in the installation. On Linux and Solaris systems, the Import/Export utility is included in the NDSadmutl package.

Novell Import Conversion Export Syntax

The Novell Import Conversion Export utility is launched with the following syntax:

```
ice general_options
-S[LDIF | LDAP | DELIM] source_options
-D[LDIF | LDAP | DELIM] destination_options
```

General options are optional and must come before any source or destination options. The -S (source) and -D (destination) handler sections can be placed in any order.

The following is a list of the available source and destination handlers:

- "LDIF Source Handler Options" on page 183
- "LDIF Destination Handler Options" on page 184
- "LDAP Source Handler Options" on page 184
- "LDAP Destination Handler Options" on page 188
- "DELIM Source Handler Options" on page 189
- "DELIM Destination Handler Options" on page 191

General Options

General options affect the overall processing of the Novell Import Conversion Export engine.

Table 36   General Options

Option	Description
-l *log_file*	Specifies a filename where output messages (including error messages) are logged. If this option is not used, error messages are sent to ice.log. If this option is not used on Linux or Solaris systems, error messages will not be logged.
-o	Overwrites an existing log file. If this flag is not set, messages are appended to the log file instead.
-e *LDIF_error_log_ file*	Specifies a filename where entries that fail are output in LDIF format. This file can be examined, modified to correct the errors, then reapplied to the directory.

Option	Description
-p *URL*	Specifies the location of an XML placement rule to be used by the engine. Placement rules let you change the placement of an entry. See "Conversion Rules" on page 193 for more information.
-c *URL*	Specifies the location of an XML creation rule to be used by the engine. Creation rules let you supply missing information that might be needed to allow an entry to be created successfully on import. For more information, see "Conversion Rules" on page 193.
-s *URL*	Specifies the location of an XML schema mapping rule to be used by the engine. Schema mapping rules let you extend the schema on the destination server to accommodate all of the object classes and attribute types in entries coming from the source server.
	You can use a schema mapping rule to map a schema element on a source server to a different but equivalent schema element on a destination server. For more information, see "Conversion Rules" on page 193.

Source Handler Options

The source handler option (-S) determines the source of the import data. Only one of the following can be specified on the command line.

Table 37   **Source Handler Options**

Option	Description
-SLDIF	Specifies that the source is an LDIF file.
	For a list of supported LDIF options, see "LDIF Source Handler Options" on page 183.
-SLDAP	Specifies that the source is an LDAP server.
	For a list of supported LDAP options, see "LDAP Source Handler Options" on page 184
-SDELIM	Specifies that the source is a comma-delimited data file.
	For a list of supported DELIM options, see "DELIM Source Handler Options" on page 189.

Destination Handler Options

The destination handler options (-D) specify the destination of the export data. Only one of the following can be specified on the command line.

Table 38   Destination Handler Options

Option	Description
-DLDIF	Specifies that the destination is an LDIF file.
	For a list of supported options, see "LDIF Destination Handler Options" on page 184.
-DLDAP	Specifies that the destination is an LDAP server.
	For a list of supported options, see "LDAP Destination Handler Options" on page 188.
-DDELIM	Specifies that the destination is a comma-delimited file.
	For a list of supported options, see "DELIM Destination Handler Options" on page 191.

LDIF Source Handler Options

The LDIF source handler reads data from an LDIF file then sends it to the Novell Import Conversion Export engine.

Table 39   LDIF Source Handler Options

Option	Description
-f *LDIF_file*	Specifies a filename containing LDIF records read by the LDIF source handler and sent to the engine.
	If you omit this option on Linux or Solaris systems, the input will be taken from stdin.
-a	If the records in the LDIF file are content records (that is, they contain no changetypes), they will be treated as records with a change type of add.

Option	Description
-c	Prevents the LDIF source handler from stopping on errors. This includes errors on parsing LDIF and errors sent back from the destination handler. When this option is set and an error occurs, the LDIF source handler reports the error, finds the next record in the LDIF file, and continues.
-n	Does not perform update operations, but prints what would be done. When this option is set, the LDIF source handler parses the LDIF file but does not send any records to the Novell Import Conversion Export engine (or to the destination handler).
-v	Enables the verbose mode of the handler.

LDIF Destination Handler Options

The LDIF destination handler receives data from the Novell Import Conversion Export engine and writes it to an LDIF file.

Table 40   **LDIF Destination Handler Options**

Option	Description
-f *LDIF_file*	Specifies the filename where LDIF records can be written. If you omit this option on Linux or Solaris systems, the output will go to stdout.
-v	Enables verbose mode of the handler.

LDAP Source Handler Options

The LDAP source handler reads data from an LDAP server by sending a search request to the server. It then sends the search entries it receives from the search operation to the Novell Import Conversion Export engine.

Table 41   LDAP Source Handler Options

Option	Description
-s *server_name*	Specifies the DNS name or IP address of the LDAP server to which the handler will send a search request. The default is the local host.
-p *port*	Specifies the integer port number of the LDAP server specified by *server_name*. The default is 389. For secure operations the default port is 636.
-d *DN*	Specifies the distinguished name of the entry that should be used when binding to the server-specified bind operation.
-w *password*	Specifies the password attribute of the entry specified by *DN*.
-W	Prompts for the password of the entry specified by *DN*. This option is applicable only for Linux and Solaris.
-F *filter*	Specifies an RFC 1558 compliant search filter. If this option is omitted, the search filter defaults to objectclass=*.
-n	Does not actually perform a search, but shows what search would be performed.
-a *attribute_list*	Specifies a comma-separated list of attributes to retrieve as part of the search. In addition to attribute names, there are three other values: • Get no attributes (1.1) • All user attributes (*) • An empty list gets all non-operational attributes. If this option is omitted, the attribute list defaults to the empty list.

Option	Description
-o *attribute_list*	Specifies a comma-separated list of attributes to be omitted from the search results received from the LDAP server before they are sent to the engine. This option is useful in cases where you want to use a wildcard with the -a option to get all attributes of some class and then remove a few of them from the search results before passing the data on to the engine. For example, -a* -o telephoneNumber searches for all user-level attributes and filters the telephone number from the results.
-R	Do not automatically follow referrals. The default is to follow referrals with the name and password given with the -d and -w options.
-e *value*	Specifies which debugging flags should be enabled in the LDAP client SDK. For more information, see "Using LDAP SDK Debugging Flags" on page 491.
-b *base_DN*	Specifies the base distinguished name for the search request. If this option is omitted, the base DN defaults to "" (empty string).
-c *search_scope*	Specifies the scope of the search request. Valid values are: ◆ One Searches only the immediate children of the base object. ◆ Base Searches only the base object entry itself. ◆ Sub Searches the LDAP subtree rooted at and including the base object. If this option is omitted, the search scope defaults to Sub.

Option	Description
-r *deref_aliases*	Specifies the way aliases should be dereferenced during the search operation. Values include:
	◆ Never
	Prevents the server from dereferencing aliases.
	◆ Always
	Causes aliases to be dereferenced when locating the base object of the search and when evaluating entries that match the search filter.
	◆ Search
	Causes aliases to be dereferenced when applying the filter to entries within the scope of the search after the base object has been located, but not when locating the base object itself.
	◆ Find
	Causes aliases to be dereferenced when locating the base object of the search, but not when actually evaluating entries that match the search filter.
	If this option is omitted, the alias dereferencing behavior defaults to Never.
-l *time_limit*	Specifies a time limit for the search in seconds.
-z *size _limit*	Specifies the maximum number of entries to be returned by the search.
-V *version*	Specifies the LDAP protocol version to be used for the connection. It must be 2 or 3. If this option is omitted, the default is 3.
-v	Enables verbose mode of the handler.
-L *filename*	Specifies a file in DER format containing a server key used for SSL authentication.
-A	Retrieves attribute names only. Attribute values are not returned by the search operation.

LDAP Destination Handler Options

The LDAP destination handler receives data from the Novell Import Conversion Export engine and sends it to an LDAP server in the form of update operations to be performed by the server.

Table 42   **LDAP Destination Handler Options**

Option	Description
-s *server_name*	Specifies the DNS name or IP address of the LDAP server to which the handler will send a search request. The default is the local host.
-p *port*	Specifies the integer port number of the LDAP server specified by *server_name*. The default is 389. For secure operations the default port is 636.
-d *DN*	Specifies the distinguished name of the entry that should be used when binding to the server-specified bind operation.
-w *password*	Specifies the password attribute of the entry specified by *DN*.
-W	Prompts for the password of the entry specified by *DN*. This option is applicable only for Linux and Solaris.
-B	Use this option if you do not want to use asynchronous LDAP Bulk Update/Replication Protocol (LBURP) requests for transferring update operations to the server. Instead, use standard synchronous LDAP update operation requests. For more information, see "LDAP Bulk Update/Replication Protocol" on page 204.
-F	Allows the creation of forward references. When an entry is going to be created before its parent exists, a placeholder called a forward reference is created for the entry's parent to allow the entry to be successfully created. If a later operation creates the parent, the forward reference is changed into a normal entry.

Option	Description
-l	Stores password values using the simple password method of the Novell Modular Authentication Service (NMAS). Passwords are kept in a secure location in the directory, but key pairs are not generated until they are actually needed for authentication between servers. This improves the speed with which an object that has password information can be loaded.
-e *value*	Specifies which debugging flags should be enabled in the LDAP client SDK. For more information, see "Using LDAP SDK Debugging Flags" on page 491.
-V *version*	Specifies the LDAP protocol version to be used for the connection. It must be 2 or 3. If this option is omitted, the default is 3.
-v	Enables verbose mode of the handler
-L *filename*	Specifies a file in DER format containing a server key used for SSL authentication.

DELIM Source Handler Options

The DELIM source handler reads data from a comma-delimited data file, then sends it to the destination handler.

Table 43   DELIM Source Handler Options

Option	Description
-f *filename*	Specifies a filename containing comma-delimited records read by the DELIM source handler and sent to the destination handler.
-F *value*	Specifies a filename containing the attribute data order for the file specified by -f. If this option is not specified, you must enter this information directly using -t. See "Performing a Comma-Delimited Import" on page 192 for more information.

Option	Description
-t *value*	Comma-delimited list of attributes specifying the attribute data order for the file specified by -f. Either this option or -F must be specified. See "Performing a Comma-Delimited Import" on page 192 for more information.
-c	Prevents the DELIM source handler from stopping on errors. This includes errors on parsing comma-delimited data files and errors sent back from the destination handler. When this option is set and an error occurs, the DELIM source handler reports the error, finds the next record in the comma-delimited data file, and continues.
-n *value*	Specifies the LDAP naming attribute for the new object. This attribute must be contained in the attribute data you specify using -F or -t.
-l *value*	Specifies the path to append the RDN to (such as o=myCompany). If you are passing the DN, this value is not necessary.
-o *value*	Comma-delimited list of object classes (if none are contained in your input file), or additional object classes such as auxiliary classes. The default value is inetorgperson.
-i *value*	Comma-delimited list of columns to skip. This value is an integer specifying the number of the column to skip. For example, to skip the third and fifth columns, specify 3,5.
-d *value*	Specifies the delimiter. The default delimiter is a comma (,).
-q *value*	Specifies the secondary delimiter. The default secondary delimiter is single-quotes (' ').

DELIM Destination Handler Options

The DELIM destination handler receives data from the source handler and writes it to a comma-delimited data file.

Table 44   **DELIM Destination Handler Options**

Option	Description
-f *filename*	Specifies the filename where comma-delimited records can be written.
-v	Enables verbose mode of the handler.
-F *value*	Specifies a filename containing the attribute data order for the source data. If this option is not specified, you must enter this information directly using -t.
-t *value*	Comma-delimited list of attributes specifying the attribute data order for the source data. Either this option or -F must be specified.
-l *value*	Can be either RDN or DN. Specifies whether the driver should place the entire DN or just the RDN in the data. RDN is the default value.
-d *value*	Specifies the delimiter. The default delimiter is a comma (,).
-q *value*	Specifies the secondary delimiter. The default secondary delimiter is single-quotes (' ').
-n *value*	Specifies a naming attribute to be appended during import, for example, cn.

Examples

Listed below are sample commands that can be used with the Novell Import Conversion Export command line utility for the following functions:

Novell eDirectory Management Utilities   **191**

DELIM Destination Handler Options

The DELIM destination handler receives data from the source handler and writes it to a comma-delimited data file.

Table 44   **DELIM Destination Handler Options**

Option	Description
-f *filename*	Specifies the filename where comma-delimited records can be written.
-v	Enables verbose mode of the handler.
-F *value*	Specifies a filename containing the attribute data order for the source data. If this option is not specified, you must enter this information directly using -t.
-t *value*	Comma-delimited list of attributes specifying the attribute data order for the source data. Either this option or -F must be specified.
-l *value*	Can be either RDN or DN. Specifies whether the driver should place the entire DN or just the RDN in the data. RDN is the default value.
-d *value*	Specifies the delimiter. The default delimiter is a comma (,).
-q *value*	Specifies the secondary delimiter. The default secondary delimiter is single-quotes (' ').
-n *value*	Specifies a naming attribute to be appended during import, for example, cn.

Examples

Listed below are sample commands that can be used with the Novell Import Conversion Export command line utility for the following functions:

Performing an LDIF Import

To perform an LDIF import, combine the LDIF source and LDAP destination handlers, for example:

```
ice -S LDIF -f entries.ldif -D LDAP -s server1.acme.com -
  p 389 -d cn=admin,c=us -w secret
```

This command line reads LDIF data from ENTRIES.LDIF and sends it to the LDAP server server1.acme.com at port 389 using the identity cn=admin,c=us, and the password "secret."

Performing an LDIF Export

To perform an LDIF export, combine the LDAP source and LDIF destination handlers, for example:

```
ice -S LDAP -s server1.acme.com -p 389 -d cn=admin,c=us -w
password -F objectClass=* -c sub -D LDIF -f server1.ldif
```

This command line performs a subtree search for all objects in the server server1.acme.com at port 389 using the identity cn=admin,c=us, and the password "password," and outputs the data in LDIF format to SERVER1.LDIF.

Performing a Comma-Delimited Import

To perform a comma-delimited import, use a command similar to the following:

```
ice -SDELIM -f/tmp/in.csv -F/tmp/order.csv -ncn -lo=acme -
DLDAP -sserver1.acme.com -p389 -dcnadmin,c=us -wsecret
```

This command reads comma-delimited values from the C:\TMP\IN.CSV file and reads the attribute order from the C:TMP\ORDER.CSV file. For each attribute entry in IN.CSV, the attribute type is specified in ORDER.CSV. For example, if IN.CSV contains the following:

```
pat,pat,engineer,john
```

Then ORDER.CSV would contain the following:

```
dn,cn,objectclass,title,sn
```

The information in ORDER.CSV could be input directly using the -t option.

The data is then sent to the LDAP server server1.acme.com at port 389 using the identity cn=admin,c=us, and password "secret."

This example specifies that cn should become the new DN for this object using the -n option, and this object was added to the organization container acme using the -l option.

Performing a Comma-Delimited Export

To perform a comma-delimited export, use a command similar to the following:

```
ice -SLDAP -sserver1.acme.com -p389 -dcn=admin,c=us
-wpassword -lobjectClass=* -csub -DDELIM -f/tmp/server1.csv -
Forder.csv
```

This command line performs a subtree search for all objects in the server server1.acme.com at port 389 using the identity cn=admin,c=us, and password "password" and outputs the data in comma-delimited format to the /TMP/SERVER1.CSV file.

Performing a Data Migration between LDAP Servers

To perform a data migration between LDAP servers, combine the LDAP source and LDAP destination handlers, for example:

```
ice -S LDAP -s server1.acme.com -p 389 -d cn=admin,c=us -w
password -F objectClass=* -c sub -D LDAP -s server2.acme.com
-p 389 -d cn=admin,c=us -w secret
```

This particular command line performs a subtree search for all objects in the server server1.acme.com at port 389 using the identity cn=admin,c=us with the password "password," and sends it to the LDAP server server2.acme.com at port 389 using the identity cn=admin,c=us and the password "secret."

Conversion Rules

The Novell Import Conversion Export engine lets you specify a set of rules that describe processing actions to be taken on each record received from the source handler and before the record is sent on to the destination handler. These rules are specified in XML (either in the form of an XML file or XML data stored in the directory) and solve the following problems when importing entries from one LDAP directory to another:

- Schema differences
- Hierarchical differences
- Missing information

There are three types of conversion rules:

Table 45   **Conversion Rules**

Rule	Description
Placement	Placement rules let you change the placement of an entry. For example, if you are importing a group of users in the l=San Francisco, c=US container but you want them to be in the l=Los Angeles, c=US container when the import is complete, you could use a placement rule to do this. For information on the format of these rules, see "Placement Rules" on page 200.
Creation	Creation rules let you supply missing information that might be needed to allow an entry to be created successfully on import. For example, assume that you have exported LDIF data from a server whose schema requires only the cn (commonName) attribute for user entries, but the server to which you are importing the LDIF data requires both the cn and sn (surname) attributes. You could use the creation rule to supply a default sn value, (such as "") for each entry as it is processed by the engine. When the entry is sent to the destination server, it will have the required sn attribute and the entry can be added successfully. For information on the format of these rules, see "Create Rules" on page 198.
Schema Mapping	When you are transferring data between servers, either directly or using LDIF, there are almost always schema differences in the servers. In some cases, you might need to extend the schema on the destination server to accommodate the object classes and attribute types in entries coming from the source server. You might also need to map a schema element on the source server to a different but equivalent schema element on the destination server. You can use schema mapping rules to do this. For information on the format of these rules, see "Schema Mapping Rules" on page 196.

You can enable conversion rules in both the Novell eDirectory Import/Export Wizard and the command line interface. For more information on XML rules, see "Using XML Rules" on page 196.

Using the Novell eDirectory Import/Export Wizard

1 In ConsoleOne, select Wizard > NDS Import/Export.

2 Click the task you want to perform.

3 Click Advanced > complete the following options:

Option	Description
Schema Rules	Specifies the location of an XML schema mapping rule to be used by the engine.
Placement Rules	Specifies the location of an XML placement rule to be used by the engine.
Creation Rules	Specifies the location of an XML creation rule to be used by the engine.

4 Click Close.

5 Follow the online instructions to finish your selected task.

Using the Command Line Interface

You can enable conversion rules with the -p, -c, and -s general options on the Novell Import Conversion Export executable. For more information, see "General Options" on page 181.

Table 46   **Conversion Rules Options**

Option	Description
-p *URL*	*URL* specifies the location of an XML placement rule to be used by the engine.
-c *URL*	*URL* specifies the location of an XML creation rule to be used by the engine.
-s *URL*	*URL* specifies the location of an XML schema mapping rule to be used by the engine.

For all three options, *URL* must be one of the following:

 ◆ A URL of the following format:

    `file://[`*path*`/]`*filename*

    The file must be on the local file system.

 ◆ An RFC 2255-compliant LDAP URL that specifies a base level search and an attribute list consisting of a single attribute description for a singled-valued attribute type.

Using XML Rules

The Novell Import Conversion Export conversion rules use the same XML format as DirXML™. For more information on DirXML, see DirXML Administration Guide (http://www.novell.com/documentation/lg/nmas20/index.html).

Schema Mapping Rules

The <attr-name-map> element is the top level element for the schema mapping rules. Mapping rules determine how the import schema interacts with the export schema. They associate specified import class definitions and attributes with corresponding definitions in the export schema.

Mapping rules can be set up for attribute names or class names.

 ◆ For an attribute mapping, the rule must specify that it is an attribute mapping, a name space (nds-name is the tag for the source name), the name in the eDirectory name space, and then the other name space (app-name is the tag for the destination name) and the name in that name space. It can specify that the mapping applies to a specific class, or it can be applied to all classes with the attribute.

 ◆ For a class mapping, the rule must specify that it is a class mapping rule, a name space (eDirectory or the application), the name in that name space, and then the other name space and the name in that name space.

The following is the formal DTD definition of schema mapping rules.

```
<!ELEMENT attr-name-map (attr-name | class-name)*>

<!ELEMENT attr-name (nds-name, app-name)>
<!ATTLIST attr-name
          class-name    CDATA    #IMPLIED>
```

```
<!ELEMENT class-name (nds-name, app-name)>

<!ELEMENT nds-name (#PCDATA)>

<!ELEMENT app-name (#PCDATA)>
```

You can have multiple mapping elements in the file. Each element is processed in the order that it appears in the file. If you map the same class or attribute more than once, the first mapping takes precedence.

The following samples illustrate how to create a schema mapping rule.

Schema Rule 1: The following rule maps the source's surname attribute to the destination's sn attribute for the inetOrgPerson class.

```
<attr-name-map>
   <attr-name class-name="inetOrgPperson">
      <nds-name>surname</nds-name>
      <app-name>sn</app-name>
   </attr-name>
</attr-name-map>
```

Schema Rule 2: The following rule maps the source's inetOrgPerson class definition to the destination's User class definition.

```
<attr-name-map>
   <class-name>
      <nds-name>inetOrgPerson</nds-name>
      <app-name>User</app-name>
   </class-name>
</attr-name-map>
```

Schema Rule 3: The following example contains two rules. The first rule maps the source's Surname attribute to the destination's sn attribute for all classes that use these attributes. The second rule maps the source's inetOrgPerson class definition to the destination's User class definition.

```
<attr-name-map>
   <attr-name>
      <nds-name>surname</nds-name>
      <app-name>sn</app-name>
   </attr-name>
   <class-name>
      <nds-name>inetOrgPerson</nds-name>
      <app-name>User</app-name>
```

```
        </class-name>
</attr-name-map>
```

Sample Command: If the schema rules are saved to an SR1.XML file, the following command instructs the utility to use the rules while processing the 1ENTRY.LDF file and to send the results to a destination file, OUTT1.LDF.

```
ice -o -sfile://sr1.xml -SLDIF -f1entry.ldf -c -DLDIF
-foutt1.ldf
```

Create Rules

Create rules specify the conditions for creating a new entry in the destination directory. They support the following elements:

- **Required Attributes:** Specifies that an add record must have values for all of the required attributes, or the add fails. The rule can supply a default value for a required attribute. If a record does not have a value for the attribute, the entry is given the default value. If the record has a value, the record value is used.

- **Matching Attributes:** Specifies that an add record must have the specific attributes and match the specified values, or the add fails.

- **Templates:** Specifies the distinguished name of a template object in eDirectory. The Novell Import Conversion Export utility does not currently support specifying templates in create rules.

The following is the formal DTD definition for create rules:

```
<!ELEMENT create-rules (create-rule)*>

<!ELEMENT create-rule (match-attr*,
                       required-attr*,
                       template?) >
<!ATTLIST create-rule
          class-name    CDATA    #IMPLIED
          description   CDATA    #IMPLIED>

<!ELEMENT match-attr    (value)+ >
<!ATTLIST match-attr
          attr-name     CDATA     #REQUIRED>

<!ELEMENT required-attr (value)*>
<!ATTLIST required-attr
          attr-name     CDATA    #REQUIRED>

<!ELEMENT template EMPTY>
```

```
<!ATTLIST template
          template-dn   CDATA   #REQUIRED>
```

You can have multiple create rule elements in the file. Each rule is processed in the order that it appears in the file. If a record does not match any of the rules, that record is skipped and the skipping does not generate an error.

The following examples illustrate how to format create rules.

Create Rule 1: The following rule places three conditions on add records that belong to the inetOrgPerson class. These records must have givenName and Surname attributes. They should have an L attribute, but if they don't, the create rule supplies a default value of Provo for them.

```
<create-rules>
   <create-rule class-name="inetOrgPerson">
      <required-attr attr-name="givenName"/>
      <required-attr attr-name="surname"/>
      <required-attr attr-name="L">
         <value>Provo</value>
      </required-attr>
   </create-rule>
</create-rules>
```

Create Rule 2: The following create rule places three conditions on all add records, regardless of their base class:

  ◆ The record must contain a givenName attribute. If it doesn't, the add fails.

  ◆ The record must contain a Surname attribute. If it doesn't, the add fails.

  ◆ The record must contain an L attribute. If it doesn't, the attribute is set to a value of Provo.

```
<create-rules>
   <create-rule>
      <required-attr attr-name="givenName"/>
      <required-attr attr-name="Surname"/>
      <required-attr attr-name="L">
         <value>Provo</value>
      </required-attr>
   </create-rule>
</create-rules>
```

Create Rule 3: The following create rule places two conditions on all records, regardless of base class:

- The rule checks to see if the record has a uid attribute with a value of ratuid. If it doesn't, the add fails.

- The rule checks to see if the record has an L attribute. If it does not have this attribute, the L attribute is set to a value of Provo.

```
<create-rules>
   <create-rule>
      <match-attr attr-name="uid">
         <value>cn=ratuid</value>
      </match-attr>
      <required-attr attr-name="L">
         <value>Provo</value>
      </required-attr>
   </create-rule>
</create-rules>
```

Sample Command: If the create rules are saved to an CRL.XML file, the following command instructs the utility to use the rules while processing the 1ENTRY.LDF file and to send the results to a destination file, OUTT1.LDF.

```
ice -o -cfile://cr1.xml -SLDIF -f1entry.ldf -c -DLDIF
-foutt1.ldf
```

Placement Rules

Placement rules determine where an entry is created in the destination directory. They support the following conditions for determining whether the rule should be used to place an entry:

- **Match Class:** If the rule contains any match class elements, an objectClass specified in the record must match the class-name attribute in the rule. If the match fails, the placement rule is not used for that record.

- **Match Attribute:** If the rule contains any match attribute elements, the record must contain an attribute value for each of the attributes specified in the match attribute element. If the match fails, the placement rule is not used for that record.

- **Match Path:** If the rule contains any match path elements, a portion of the record's dn must match the prefix specified in the match path element. If the match fails, the placement rule is not used for that record.

The last element in the rule specifies where to place the entry. The placement rule can use zero or more of the following:

- **PCDATA:** Uses parsed character data to specify the DN of a container for the entries.

- **Copy the Name:** Specifies that the RDN of the old DN is used in the entry's new DN.

- **Copy the Attribute:** Specifies the naming attribute to use in the entry's new DN. The specified naming attribute must be a valid naming attribute for the entry's base class.

- **Copy the Path:** Specifies that the source DN should be used as the destination DN.

- **Copy the Path Suffix:** Specifies that the source DN, or a portion of its path, should be used as the destination DN. If a match-path element is specified, only the part of the old DN that matches the prefix attribute of the match-path element is used as part of the entry's DN.

The following is the formal DTD definition for the placement rule.

```
<!ELEMENT placement-rules (placement-rule*)>
<!ATTLIST placement-rules
        src-dn-format    (%dn-format;)    "slash"
        dest-dn-format   (%dn-format;)    "slash"
        src-dn-delims    CDATA            #IMPLIED
        dest-dn-delims   CDATA            #IMPLIED>

<!ELEMENT placement-rule (match-class*,
                          match-path*,
                          match-attr*,
                          placement)>
<!ATTLIST placement-rule
        description      CDATA            #IMPLIED>

<!ELEMENT match-class     EMPTY>
<!ATTLIST match-class
        class-name       CDATA            #REQUIRED>

<!ELEMENT match-path      EMPTY>
<!ATTLIST match-path
        prefix           CDATA            #REQUIRED>

<!ELEMENT match-attr      (value)+ >
<!ATTLIST match-attr
        attr-name        CDATA            #REQUIRED>

<!ELEMENT placement       (#PCDATA |
                          copy-name |
                          copy-attr |
                          copy-path |
```

```
                                  copy-path-suffix)* >
```

You can have multiple placement-rule elements in the file. Each rule is processed in the order that it appears in the file. If a record does not match any of the rules, that record is skipped, and the skipping does not generate an error.

The following examples illustrate how to format placement rules. The scr-dn-format="ldap" and dest-dn-format="ldap" attributes set the rule so that the name space for the dn in the source and destination is LDAP format.

The Novell Import Conversion Export utility only supports source and destination names in LDAP format.

Placement Example 1: The following placement rule requires that the record have a base class of inetOrgPerson. If the record matches this condition, the entry is placed immediately subordinate to the test container and the left-most component of its source dn is used as part of its dn.

```
<placement-rules src-dn-format="ldap" dest-dn-format="ldap">
    <placement-rule>
      <match-class class-name="inetOrgPerson"></match-class>
       <placement>o=test<copy-name/></placement>
    </placement-rule>
 </placement-rules>
```

With this rule, a record with a base class of inetOrgPerson and with the following dn:

```
dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ
```

would have the following dn in the destination directory:

```
dn: cn=Kim Jones, o=test
```

Placement Example 2: The following placement rule requires that the record have an sn attribute. If the record matches this condition, the entry is placed immediately subordinate to the test container and the left-most component of its source dn is used as part of its dn.

```
<placement-rules src-dn-format="ldap" dest-dn-format="ldap">
    <placement-rule>
        <match-attr attr-name="sn"></match-attr>
        <placement>o=test<copy-name/></placement>
    </placement-rule>
 </placement-rules>
```

With this rule, a record with the following dn and sn attribute:

```
dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ
sn: Jones
```

would have the following dn in the destination directory:

```
dn: cn=Kim Jones, o=test
```

Placement Example 3: The following placement rule requires the record to have an sn attribute. If the record matches this condition, the entry is placed immediately subordinate to the test container and its sn attribute is used as part of its dn. The specified attribute in the copy-attr element must be a naming attribute of the entry's base class.

```
<placement-rules src-dn-format="ldap" dest-dn-format="ldap">
  <placement-rule>
    <match-attr attr-name="sn"></match-attr>
    <placement>o=test<copy-attr attr-name="sn"/></placement>
  </placement-rule>
</placement-rules>
```

With this rule, a record with the following dn and sn attribute:

```
dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ
sn: Jones
```

would have the following dn in the destination directory:

```
dn: cn=Jones, o=test
```

Placement Sample 4: The following placement rule requires the record to have an sn attribute. If the record matches this condition, the source dn is used as the destination dn.

```
<placement-rules src-dn-format="ldap" dest-dn-format="ldap">
    <placement-rule>
       <match-attr attr-name="sn"></match-attr>
       <placement><copy-path/></placement>
    </placement-rule>
 </placement-rules>
```

Placement Sample 5: The following placement rule requires the record to have an sn attribute. If the record matches this condition, the entry's entire DN is copied to the test container.

```
<placement-rules src-dn-format="ldap" dest-dn-format="ldap">
    <placement-rule>
        <match-attr attr-name="sn"></match-attr>
        <placement>o=test<copy-path-suffix/></placement>
    </placement-rule>
 </placement-rules>
```

With this rule, a record with the following dn and sn attribute:

```
dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ
sn: Jones
```

would have the following dn in the destination directory:

```
dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ, o=test
```

Sample Command: If the placement rules are saved to an PR1.XML file, the following command instructs the utility to use the rules while processing the 1ENTRY.LDF file and to send the results to a destination file, FOUTT1.LDF

```
ice -o -pfile://pr1.xml -SLDIF -f1entry.ldf -c -DLDIF
-foutt1.ldf
```

LDAP Bulk Update/Replication Protocol

The Novell Import Conversion Export utility uses the LDAP Bulk Update/ Replication Protocol (LBURP) to send asynchronous requests to an LDAP server. This guarantees that the requests are processed in the order specified by the protocol and not in an arbitrary order influenced by multiprocessor interactions or the operating system's scheduler.

LBURP also lets the Novell Import Conversion Export utility send several update operations in a single request and receive the response for all of those update operations in a single response. This adds to the network efficiency of the protocol.

LBURP works as follows:

1. The Novell Import Conversion Export utility binds to an LDAP server.

2. The server sends a bind response to the client.

3. The client sends a start LBURP extended request to the server.

4. The server sends a start LBURP extended response to the client.

5. The client sends zero or more LBURP operation extended requests to the server.

   These requests can be sent asynchronously. Each request contains a sequence number identifying the order of this request relative to other requests sent by the client over the same connection. Each request also contains one or more LDAP update operations.

6. The server processes each of the LBURP operation extended requests in the order specified by the sequence number and sends an LBURP operation extended response for each request.

7. After all of the updates have been sent to the server, the client sends an end LBURP extended request to the server.

8. The server sends an end LBURP extended response to the client.

The LBURP protocol lets Novell Import Conversion Export present data to the server as fast as the network connection between the two will allow. If the network connection is fast enough, this lets the server stay busy processing update operations 100% of the time because it never has to wait for Novell Import Conversion Export to give it more work to do.

The LBURP processor in eDirectory also commits update operations to the database in groups to gain further efficiency in processing the update operations. LBURP can greatly improve the efficiency of your LDIF imports over a traditional synchronous approach.

LBURP is enabled by default, but you can choose to disable it during an LDIF import.

To enable or disable LBURP during an LDIF import:

1 In ConsoleOne, select Wizard > NDS Import/Export.

2 Click Import LDIF File > Next.

3 Enter the name of the LDIF file containing the data you want to import > click Next.

4 Select the LDAP server where the data will be imported.

5 Click Advanced > click Use LBURP.

6 Follow the online instructions to complete the remainder of the LDIF import wizard.

IMPORTANT: Since LBURP is a relatively new protocol, eDirectory servers prior to version 8.5 (and most non-eDirectory servers) do not support it. If you are using the Novell eDirectory Import/Export Wizard to import an LDIF file to one of these servers, you must disable the LBURP option for the LDIF import to work.

You can use the command line option to enable or disable LBURP during an LDIF import. To do so, use the option "-B" on page 188.

Migrating Schema Between LDAP Directories

You can refer to Application Notes (http://www.developer.novell.com/research/) on the Novell Developer Portal for more information about migrating schema between LDAP directories.

Improving the Speed of LDIF Imports

In cases where you have thousands or even millions of records in a single LDIF file you are importing, consider the following:

Importing Directly to a Server with a Read/Write Replica

If it's possible to do so, select a destination server for your LDIF import that has read/write replicas containing all the entries represented in the LDIF file. This will maximize network efficiency.

Avoid having the destination server chain to other eDirectory servers for updates. This can severely reduce performance. However, if some of the entries to be updated are only on eDirectory servers that are not running LDAP, you might have to allow chaining to import the LDIF file.

For more information on replicas and partition management, see Chapter 6, "Managing Partitions and Replicas," on page 161.

Using LBURP

Novell Import Conversion Export maximizes network and eDirectory server processing efficiency by using LBURP to transfer data between the wizard and the server. Using LBURP during an LDIF import greatly improves the speed of your LDIF import.

For more information on LBURP, see "LDAP Bulk Update/Replication Protocol" on page 204.

Configuring the Database Cache

The amount of database cache available for use by eDirectory has a direct bearing on the speed of LDIF imports, especially as the total number of entries on the server increases. When doing an LDIF import, you might want to allocate the maximum memory possible to eDirectory during the import. After the import is complete and the server is handling an average load, you can restore your previous memory settings. This is particularly important if the import is the only activity taking place on the eDirectory server.

See Chapter 13, "Maintaining Novell eDirectory," on page 431 for more information on configuring the eDirectory database cache.

Using Simple Passwords

eDirectory uses public and private key pairs for authentication. Generating these keys is a very CPU-intensive process. With eDirectory 8.6, you can choose to store passwords using the simple password feature of the Novell Modular Authentication Service™ (NMAS™). When you do this, passwords are kept in a secure location in the directory, but key pairs are not generated until they are actually needed for authentication between servers. This greatly improves the speed with which an object that has password information can be loaded.

To enable simple passwords during an LDIF import:

1 In ConsoleOne, click Wizard > NDS Import/Export.

2 Click Import LDIF File > Next.

3 Enter the name of the LDIF file containing the data you want to import > click Next.

4 Select the LDAP server where the data will be imported.

5 Click Advanced > click Store NMAS Simple Passwords/Hashed Passwords.

6 Follow the online instructions to complete the remainder of the Wizard.

If you choose to store passwords using simple passwords, you must use an NMAS-aware Novell Client™ to log in to the eDirectory tree and access traditional file and print services. NMAS must also be installed on the server. LDAP applications binding with name and password will work seamlessly with the simple password feature.

For more information on NMAS, see *Novell Modular Authentication Services Installation and Administration Guide* (http://www.novell.com/documentation/lg/nmas20/index.html).

Using Indexes Appropriately

Having unnecessary indexes can slow down your LDIF import because each defined index requires additional processing for each entry having attribute values stored in that index. You should make sure that you don't have unnecessary indexes before you do an LDIF import, and you might want to consider creating some of your indexes after you have finished loading the data reviewed predicate statistics to see where they are really needed.

See "Index Manager" on page 231 for more information on tuning indexes.

Novell iMonitor

Novell iMonitor provides cross-platform monitoring and diagnostic capability to all servers in your eDirectory tree. This utility lets you monitor your servers from any location on your network where a Web browser is available.

iMonitor lets you look at the eDirectory environment in depth on a partition, replica, or server basis. You can also examine what tasks are taking place, when they are happening, what their results are, and how long they are taking.

iMonitor provides a Web-based alternative or replacement for many of Novell's traditional server-based eDirectory tools such as DSBROWSE, DSTRACE, DSDIAG, and the diagnostic features available in DSREPAIR. Because of this, iMonitor's features are primarily server-focused, meaning that they focus on the health of individual eDirectory agents (running instances of the directory service) rather than the entire eDirectory tree.

iMonitor 1.5 provides the following features:

- ◆ eDirectory health summary
    - ◆ Synchronization information
    - ◆ Known servers
    - ◆ Agent configuration
- ◆ eDirectory health checks
- ◆ Hyperlinked DS Trace
- ◆ Agent configuration
- ◆ Agent activity and verb statistics
- ◆ Reports
- ◆ Agent information
- ◆ Error information
- ◆ Object/schema browser
- ◆ DirXML monitor
- ◆ Search
- ◆ Partition list
- ◆ Agent process status
- ◆ Background process schedule
- ◆ DS Repair

The information you can view in iMonitor is based the following factors:

- ◆ The identity you have established

    Your identity's eDirectory rights are applied to every request you make in iMonitor. For example, you must log in as the Administrator of the server or a console operator on the server where you are trying to access the DS Repair page.

- ◆ The eDirectory agent version you are monitoring

    Newer versions of NDS and eDirectory will have features and options that older versions do not.

The information you view in iMonitor immediately shows what is happening on your server.

System Requirements

To use iMonitor 1.5, you need:

- Any HTML 3 browser with CSS, such as, Netscape 4.06 or later or Internet Explorer 4 or later, that supports CSS

- Novell eDirectory 8.5 or later

Platforms

The iMonitor 1.5 utility runs on the following platforms:

- NetWare 5 Support Pack 4 or later (for SSL, NetWare 5.1 or later)

  Novell iMonitor is placed in AUTOEXEC.NCF.

- Windows* NT*/2000 (No SSL)

- Linux*

- Solaris*

For Windows NT/2000, Linux, and Solaris, iMonitor loads automatically when eDirectory runs.

eDirectory Versions That Can Be Monitored

You can use iMonitor to monitor the following versions of NDS and eDirectory:

- All versions of NDS and eDirectory for NetWare 4.11 or later

- All versions of NDS and eDirectory for Windows NT/2000

- All versions of NDS and eDirectory for Linux, Solaris

Accessing iMonitor

To access iMonitor:

1 Ensure that the iMonitor executable is running on the eDirectory server.

2 Open your Web browser.

3 In the address (URL) field, enter:

http://*server's_TCPIP_address:httpstack_port*/nds

for example:

```
http://137.65.135.150:8008/nds
```

DNS names can be used anywhere a server's IP or IPX™ address or distinguished name could be used in iMonitor. For example, when you have configured DNS, then:

```
http://prv-gromit.provo.novell.com/nds?server=prv-
igloo.provo.novell.com
```

is equivalent to

```
http://prv-gromit.provo.novell.com/nds?server=IP_or_IPX
address
```

or

```
http://prv-gromit.provo.novell.com/nds?server=/cn=prv-
igloo,ou=ds,ou=dev,o=novell,t=novell_inc
```

4 To have access to all of the features, click Login.

Log in as Administrator with the fully distinguished name or administrator equivalent.

iMonitor Architecture

This section contains information about the following:

- "Anatomy of an iMonitor Page" on page 212
- "Modes of Operation" on page 213
- "iMonitor Features on Every Page" on page 215
- "NetWare Remote Manager Integration" on page 215
- "Configuration Files" on page 216

Anatomy of an iMonitor Page

Each iMonitor page is divided into four frames or sections: the Navigator frame, the Assistant frame, the Data frame, and the Replica frame.

Figure 26   iMonitor Frames
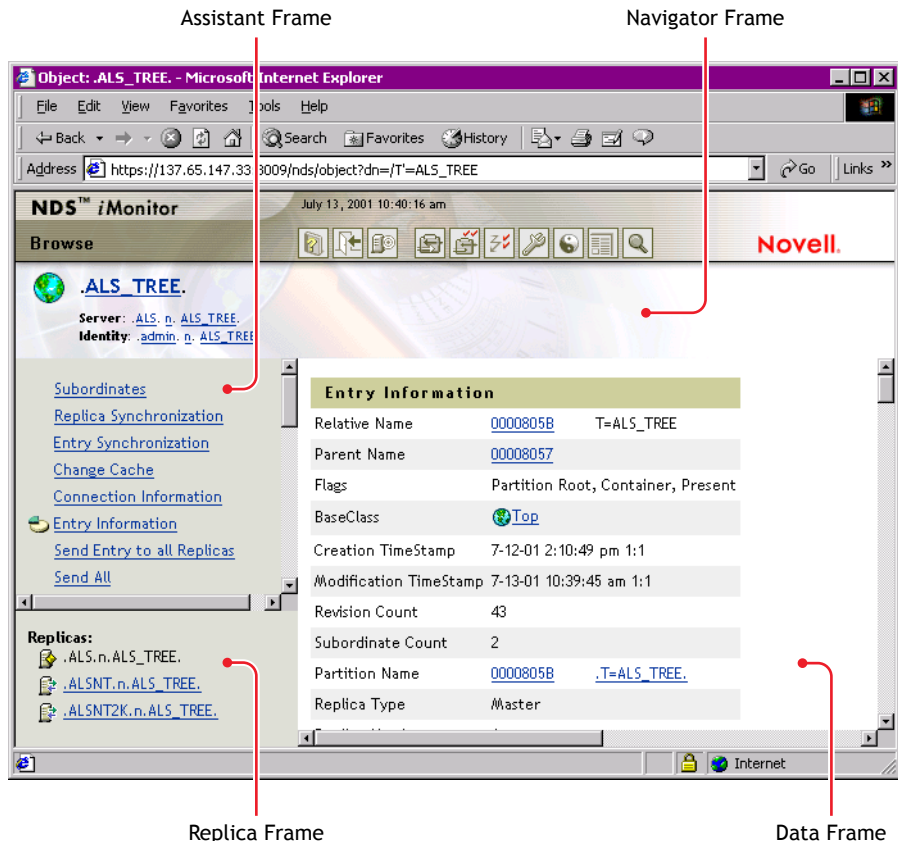
Navigator Frame: The Navigator frame is located across the top of the page. This frame shows the server name where the data is being read from, your identity, and the icons you can click to link to other screens, including online help, login, server portal, and other iMonitor pages.

Assistant Frame: The Assistant frame is located at the left side of the page. This frame contains additional navigational aids, such as links to other pages, items that help you navigate data in the Data frame, or other items to assist you with obtaining or interpreting the data on a given page.

Data Frame: The Data frame shows the detailed information about your servers that you request by clicking one of the links listed above. This is the only page you will see if your Web browser does not support frames.

Replica Frame: The Replica frame lets you determine which replica you are currently viewing, and provides links to view the same information from another replica or server's point of view. This frame only appears when you view pages where another replica of the requested data exists or where another replica may have a different view of the information being presented in the Data frame.

Modes of Operation

Novell iMonitor can be used in two different modes of operation: Direct mode and Proxy mode. No configuration changes are necessary to move between these modes. Novell iMonitor will move between these modes automatically, but you should understand them in order to successfully and easily navigate the eDirectory tree.

Figure 27   Modes of Operation
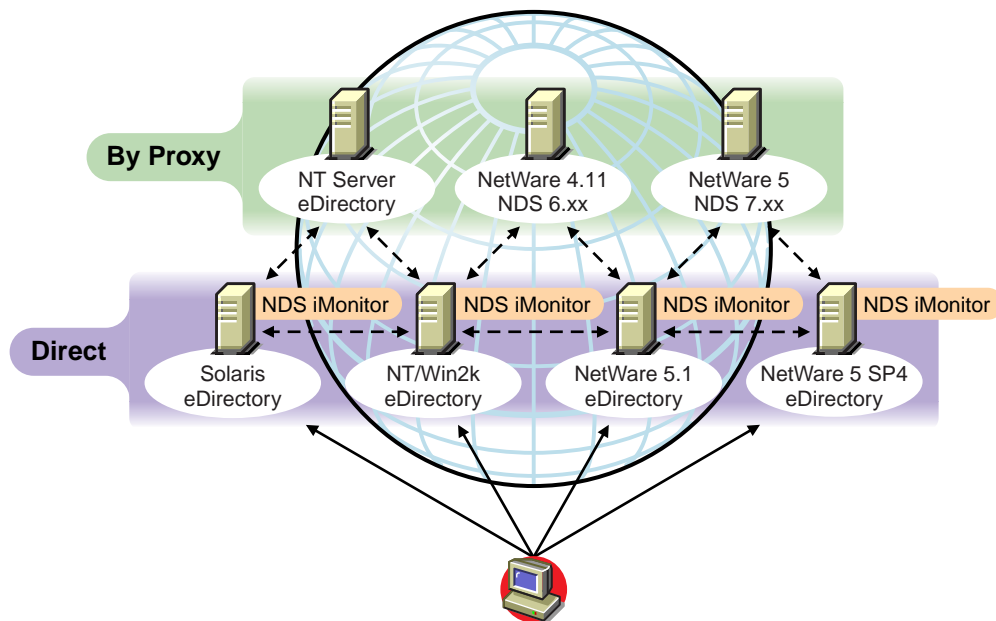
Direct Mode: Use Direct mode when your Web browser is pointed directly at an address or DNS name on a machine running the iMonitor executable and reading information only on that machine's local eDirectory DIB.

Some iMonitor features are server-centric; that is, they are only available to the iMonitor running on that machine. These features use local API sets that cannot be accessed remotely. Server-centric features in iMonitor include the DS Trace, DS Repair, and Background Process Schedule pages. When using direct mode, all iMonitor features will be available on that machine.

Key features of direct mode:

- Full server-centric feature set

- Reduced network bandwidth (faster access)

- Access by proxy still available for all versions of eDirectory

Proxy Mode: Use Proxy mode when your Web browser is pointed at an iMonitor running on one machine, but is gathering information from another machine. Because iMonitor uses traditional eDirectory non-server-centric protocols for non-server-centric features, all previous versions of eDirectory beginning with NDS 6.*x* can be monitored and diagnosed. However, server-centric features use APIs that cannot be accessed remotely.

While in proxy mode, if you wish to switch to Direct mode for a different server, you can do so as long as it is a version of eDirectory in which iMonitor has shipped. If the server you are gathering information on by proxy has iMonitor running, you will see an additional icon button in the Navigator frame. When you mouse over the icon, you will see a link to the remote iMonitor on the remote server. If the server you are gathering information on by proxy is an earlier version of eDirectory, no additional icon will be shown and you will always have to gather information on that server by proxy until it is upgraded to a version of eDirectory that includes iMonitor.

Key features of proxy mode:

- Not every server in the tree must be running Novell iMonitor in order to use most iMonitor features.

- Only one server is required to be upgraded.

- There is a single point of access for dial-in.

- You can access iMonitor over a slower speed link while iMonitor accesses eDirectory information over higher speed links.

- Previous NDS version information is accessible.

- Server-centric features are only available where iMonitor is installed.

iMonitor Features on Every Page

You can link to the Agent Summary, Agent Information, Agent Configuration, Trace Configuration, DS Repair, Reports, and Search pages from any iMonitor page by using the icons in the Navigator frame. You can also log in or link to the Novell Support Connection™ Web page from any iMonitor page.

Login/Logout: The Login button is available if you are not logged in. A Logout button, which closes your browser window, is displayed if you are logged in. Unless all browser windows are closed, your iMonitor session remains open, and you will not need to log in again. You can see your login status on any page by looking at Identity in the Navigator frame.

Support Connection Link: The Novell logo in the upper-right corner is a link to the Novell Support Connection® Web page. This provides a direct link to the Novell Web site for current server patch kits, updates, and product-specific support.

NetWare Remote Manager Integration

On NetWare 5.*x* and later servers, a link to NetWare Remote Manager is available to provide you with Web-based monitoring, diagnosis, and troubleshooting information for NetWare servers.

iMonitor is integrated with NetWare Remote Manager in the following ways:

- NetWare Remote Manager's lightweight Web server (HTTPSTK.NLM) provides the first layer of the iMonitor architecture on the NetWare platform.

- iMonitor registers with NetWare Remote Manager (PORTAL.NLM) so that links to iMonitor and other eDirectory-specific information are available through the NetWare Remote Manager interface.

  These links can be found under the Manage eDirectory section in the Remote Manager interface. Links to eDirectory agent health information can also be found in the Diagnose Server section under Health Monitor in the eDirectory-related categories.

NetWare Remote Manager also registers with eDirectory, which allows iMonitor and NetWare Remote Manager to cross-reference each other for a more seamless movement between each tool.

Configuration Files

A configuration file is included with iMonitor to allow you to change or set default behavior or values in the utility.

The configuration file is a text file containing configuration parameter tags together with their desired values. This file is located in the same directory as the iMonitor executable (which is usually in the same location as the Novell eDirectory executables).

Table 47   **iMonitor Configuration Files**

Server	Configuration File
NetWare, Windows NT, and Windows 2000	NDSIMON.INI
Solaris and Linux	ndsimon.conf

There are two groups of parameters that you can set in iMonitor's configuration file.

- ◆ Parameters that apply to how the iMonitor executable itself runs

  Except on NetWare, when the iMonitor executable loads, it will attempt to listen on the traditional HTTP port 80. If that port is in use, it will back off to port 8008. If that port is in use, iMonitor will then back off, increasing the desired port by 2 (8010, 8012, and so on) until 8078.

  Where SSL is configured and available, a similar bind pattern is attempted. First, port 81 is tried, and then 8009, 8011, 8013, and so on.

  This allows iMonitor to coexist with a Web server running on the same server. However, on some platforms, iMonitor may load before the installed Web server does, or you might want iMonitor to bind to a port of your choice. Both regular and SSL ports can be configured using the HttpPort and the HttpsPort parameters respectively. Commented out examples exist in the shipping configuration file. By default, iMonitor binds to all NIC addresses on the server where it loads. However, there is an Address parameter which you can use to specify a list of addresses, in comma delimited format, to bind to.

  On NetWare, similar port selection rules are used, but they are controlled by the NetWare Remote Manager HTTP stack (HTTPSTK.NLM) and work as specified in the NetWare Remote Manager documentation.

◆ Parameters that apply to specific features or pages

The configuration file that ships with iMonitor contains samples of the parameters that can be modified. These parameters are preceded by a # character. This indicates that they are commented out or not used when iMonitor parses the configuration file. For the shipping configuration file, iMonitor uses all internally bound default values for these parameters. To enable any of these parameters or to add any parameters, simply omit the # character from the beginning of the line.

iMonitor Features

This section provides a brief description of iMonitor features.

Online help is provided in each section of iMonitor for more detailed information about each feature and function.

Viewing eDirectory Server Health

From the Agent Summary page you can view the health of your eDirectory servers, including synchronization information, agent process status, and the total servers known to your database.

1 In iMonitor, click Agent Summary .

2 Choose from the following options:

Agent Synchronization Summary: You can view the number and types of replicas you have and the length of time since they have been successfully synchronized. You can also view the number of errors for each replica type. If there is only one replica or partition to view, the heading is Partition Synchronization Status.

If the Agent Synchronization Summary doesn't appear, there are no replicas you can view based on your identity.

Servers Known to Database Totals: You can view the type and count of servers known to your database, and whether they are up or down.

Agent Process Status Totals: You can view the status of processes without the administrator's intervention that run on an agent. When there is a problem or piece of information, a status is recorded. The table increases or decreases, depending on the number of recorded statuses.

Viewing Partition Synchronization Status

From the Agent Synchronization page you can view the synchronization status of your partitions. You can filter the information by selecting from the options listed in the Assistant frame on the left side of the page.

1 In iMonitor, click Agent Synchronization in the Assistant frame.

2 Choose from the following options:

Partition Synchronization Status: You can view the partition, number of errors, last successful synchronization, and maximum ring delta.

Partition: For each partition, you can view the links to that partition's Replica Synchronization page.

Last Successful Sync: You can view the amount of time since all replicas of an individual partition were successfully able to synchronize from the server.

Maximum Ring Delta: The maximum ring delta shows the amount of data which might not be successfully synchronized to all the replicas in the ring. For example, if a user has changed his login script within the past 30 minutes, and the maximum ring delta has a 45-minute allocation, the user's login might not be successfully synchronized, and he might get the previous login script when he attempts to log in. If, however, the user changed his login script more than 45 minutes ago, he should get the new login script consistently from all replicas.

If Unknown is listed under Maximum Ring Delta, it means the transitive synchronized vector is inconsistent and the maximum ring delta cannot be calculated due to replica/partition operations occurring, or some other problem.

Viewing Server Connection Information

From the Agent Information page you can view the connection information for your server.

1 In iMonitor, click Agent Information in the Assistant frame.

2 Choose from the following options:

Ping Info: Depending on the transport, configuration, and platform you are running on, you might not see ping information. The information, if listed, shows that iMonitor has attempted an IP ping to the set of addresses being advertised for the server. Success is as indicated.

DNS Name: Depending on the transport, configuration, and platform you are running on, you might not see this information. The information, if listed, shows that iMonitor has attempted to do an address reversal on IP addresses supported by the server and is indicating the associated DNS name.

Connection Information: You can view connection information for the server, including the server referral, time delta, Root Most Master, and replica depth.

Server Referral: You can view the set of addresses by which your server can be reached.

Time Synchronized: eDirectory believes time is synchronized well enough to issue time stamps based on the server's current time. The time synchronization protocol might or might not currently be in a synchronized state. Time Synchronized indicates that synthetic or future time is not being used unless a replica's last-issued time stamp is greater than the current time.

Time Delta: You can view the difference in time between iMonitor and the remote server in seconds. A negative integer indicates that iMonitor's time is ahead of the server's time; a positive integer indicates that iMonitor's time is slower than the server.

Root Most Master: Root Most Master specifies that the replica that is highest or closest to the root of the naming tree is a master replica.

Replica Depth: You can view the depth of the Root Most Replica (the number of levels between the Root Most Replica and the root of the tree).

Viewing Known Servers

From the Known Servers List, you can view the list of servers known to the database of the source server. You can filter the list to show all servers known to the database or to show all servers in the replica ring. If a server has an icon next to it, the server participates in a replica ring.

1 In iMonitor, click Known Servers in the Assistant frame.

2 Choose from the following options:

Entry ID: The Entry ID column lists the identifier on the local server for an object. Entry IDs cannot be used across servers.

NDS Revision: The NDS Revision column lists the eDirectory build number or version being cached or stored on the server with which you're communicating.

Status: The Status column shows whether the server is up, down, or unknown. If the status shows as unknown, this means the server has never needed to communicate with the server being shown as unknown.

Last Updated: The Last Updated column shows the last time this server attempted to communicate with the server and found out it was down. If this column is not showing, all servers are currently up.

Viewing Replica Information

From the Partitions page you can view information about the replicas on the server you are communicating with. You can filter the page by selecting from the options in the Assistant frame on the left side of the page.

Server Partition Information: You can view information about the server's partition, including the entry ID, replica state, purge time, and last modification time.

Partition: You can view information about the partition Tree object on the server.

Purge Time: Data that has been deleted before the listed purge time can be removed from the database because all replicas have seen the deletion.

Last Modification Time: You can view the last-issued time stamp of data written to the database for the replica. This lets you see if time is in the future and if synthetic time is being used.

Replica Synchronization: You can click the Replica Synchronization link to view the Replica Synchronization Summary page that refers to the partition. The Replica Synchronization page shows information about the partition synchronization status and replica status. You can also view lists of partitions and replicas.

Controlling and Configuring the DS Agent

From the Agent Configuration page you can control and configure the DS Agent. The functionality you have on this page will depend on the rights of the current identity and the version of eDirectory you are looking at.

1 In iMonitor, click Agent Configuration .

2 Choose from the following options:

Agent Triggers: You can use agent triggers to initiate certain background processes. These triggers are equivalent to using the SET DSTRACE=*option* command.

Background Process Settings: You can use background process settings to modify the interval at which certain background processes run. These settings are equivalent to the SET DSTRACE=!*option* command.

Agent Synchronization: You can use agent synchronization settings to disable or enable inbound or outbound synchronization. You can specify in hours the amount of time you want synchronization disabled.

Database Cache: You can configure the amount of database cache used by the DS database engine. Various cache statistics are also provided to assist you in determining whether you have an appropriate amount of cache available. Having an inadequate amount of cache may severely impact your system's performance.

Configuring Trace Settings

Use the Trace Configuration page to set trace settings. Novell iMonitor's DS Trace is a server-centric feature. That is, it can only be initiated on a server where iMonitor is running. If you need to access this feature on another server, you must switch to the iMonitor running on that server.

To access information on the Trace Configuration page you must be the equivalent of Administrator of the server or a console operator. You are prompted to enter your username and password so your credentials can be verified before you can access information on this page.

1 In iMonitor, click Trace Configuration .

2 Choose from the following options:

Submit: You can submit changes to Trace Options and Trace Line Prefixes. If DS Trace is off, click Submit to turn it on. If DS Trace is already on, click Submit to submit changes to the current trace.

Trace On/Off: You can turn DS Trace on or off by using this button. The button text will change based on the current DS Trace state. If DS Trace is on, the button text will say Trace Off. Clicking it turns DS Trace off and vice versa. When DS Trace is off, clicking Trace On is equivalent to clicking Submit.

DS Trace Options: These options apply to the events on the local DS Agent where the trace is initiated. The options show errors, potential problems, and other information about eDirectory on your local server. Turning on DS Trace options can increase CPU utilization and might reduce your system's performance; therefore, DS Trace should generally be used for diagnostic purposes, not as a standard practice. These options are a more convenient equivalent of the SET DSTRACE=+*option* command.

Trace Line Prefixes: You can choose which pieces of data are added to the beginning of any trace line. All trace line prefixes are selected by default.

Trace History: You can view a list of previous trace runs. Each previous trace log is identified by the period of time during which the trace data was being gathered.

Trace Triggers: You can use trace triggers to show the trace flags that must be set in order to display the specified DS Agent information in DS Trace. These triggers might write large quantities of information to trace. Generally, we recommend that these triggers be enabled only when instructed by Novell Technical Support.

3 Click Submit to submit any changes.

4 Click Trace On to turn DS Trace on > click to view DS Trace in iMonitor.

Viewing Process Status Information

From the Agent Process Status page you can view background process status errors and more information about each error that occurred. You can filter the information on this page by selecting from the options listed in the Assistant frame on the left side of the page.

1 In iMonitor, click Agent Process Status in the Assistant frame.

Background process statuses that are currently reported include:

- ◆ Schema synchronization
- ◆ Obituary processing
- ◆ External reference/DRL
- ◆ Limber

Viewing Agent Activity

From the Agent Activity page you can determine traffic patterns and potential system bottlenecks. You can use this page to view the verbs and requests that are currently being handled by eDirectory. You can also see which of those requests are attempting to obtain DIB locks in order to write to the database and how many of those requests are waiting to obtain a DIB lock.

If you are viewing a server running Novell eDirectory 8.6, you will also see a list of partitions and the servers that participate in the replica ring with the server specified in the Navigator frame. With the introduction of Novell eDirectory 8.6, synchronization is no longer single threaded. Any 8.6 server might outbound multiple partitions simultaneously to one or more replication partners. For this reason, the synchronization activity page was created so you can more easily monitor this parallel synchronization strategy.

1 In iMonitor, click Agent Activity in the Assistant frame.

2 Choose from the following options:

Verb Statistics: Lets you view a running count of all verbs called and requests made since NDS was last initialized. This page also shows how many of those requests are currently active and the minimum, maximum, and average times (shown in milliseconds) that it takes to process those requests.

Background Process Schedule: Lets you view the background processes that are scheduled, what their current state is, and when they are scheduled to run again.

Synchronization Activity: Displays a list of different times that inbound and outbound synchronization occurred. If inbound or outbound synchronization is currently taking place, you will see an icon indicating that the process is active, when that cycle was started, and with which server it is occurring.

If inbound and outbound synchronization is disabled, you will see an icon indicating that fact and when it is scheduled to be reenabled. For outbound synchronization, the next scheduled time is also shown.

Viewing Traffic Patterns

From the Verb Statistics page you can determine traffic patterns and potential system bottlenecks. You can use this page to view a running count of all verbs called and requests made since eDirectory was last initialized. This page also shows how many of those requests are currently active and the minimum, maximum, and average times (shown in milliseconds) it takes to process those requests. Background process, bindery, and standard eDirectory requests are tracked.

If you view this page on an older version of eDirectory, you might not see as much information as you would if you were running on DS build number 8500 or later.

Viewing Background Processes

From the Background Process Schedule page you can view the background processes that are scheduled, what their current state is, and when they are scheduled to run again. Novell iMonitor's Background Process Schedule is a server-centric feature. That is, it can only be viewed on a server where iMonitor is running. If you need to access the background process schedule on another server, you must switch to the iMonitor running on that server. As you upgrade more servers to eDirectory 8.5, iMonitor's server-centric features will be more available to you. Other server-centric features include the DS Trace and DS Repair pages.

To access information on the Background Process Schedule page, you must be the equivalent of Administrator of the server or a console operator. You are prompted to log in so your credentials can be verified before you can access information on this page.

Viewing eDirectory Server Errors

From the Error Index page you can view information about the errors found on your eDirectory servers. The errors are separated into two fields: eDirectory-specific errors and other errors that might be of interest. Each error listed is hyperlinked to a description that contains an explanation, possible cause, and troubleshooting actions.

1 In iMonitor, click Error Index in the Assistant frame.

From the Error Index page you can link to the latest Novell documentation on errors, technical information, and white papers.

Viewing DS Repair Information

From the DS Repair page you can view problems and back up or clean up your DIB sets. Novell iMonitor's DS Repair is a server-centric feature. That is, it can only be initiated on a server where iMonitor is running. If you need to access the DS Repair information on another server, you must switch to the iMonitor running on that server. As you upgrade more servers to later versions of eDirectory, iMonitor's server-centric features will be more available to you. Other server-centric features include the DS Trace and Background Process Schedule pages.

To access information on this page, you must be the equivalent of Administrator of the server or a console operator. You are prompted to log in so your credentials can be verified before you can access information on this page.

1 In iMonitor, click DS Repair .

2 Choose from the following options:

Downloads: Retrieve repair-related files from the file server. You will not be able to access DSREPAIR.LOG if the DSREPAIR utility is running or you have initiated a repair from the DS Repair page in iMonitor until the operation is finished.

Delete Old DIB Sets: Click the red X to delete an old DIB set.

WARNING: This action is irreversible. When you select this option, the old DIB set will be purged from the file system.

DS Repair Advanced Switches: Use the advanced switches to fix problems, check for problems, or create a backup of your database. You will not need to enter information in the Support Options field unless you are directed to do so by Novell Technical Support.

3 Click Start Repair to run DS Repair on this server.

Viewing Agent Health Information

Use the Agent Health page to view health information about the specified eDirectory agent and the partitions and replica rings it participates in.

1 In iMonitor, click Agent Health in the Assistant frame.

2 Click the links to view detailed information.

Browsing Objects in Your Tree

The Browse page let you browse any object in your tree. The Navigation bar at the top of the page lets you know what server the object you are viewing is on, and the path to the object. The Replica frame on the left of the page lets you view or access the same object on any real partition. Click any underlined object on the page to view more information about an object. You can also click any portion of the name in the Navigator frame to browse up the tree.

The information displayed on this page depends on the eDirectory rights you are logged in with, the type of object you are browsing, and the version of NDS or eDirectory you are running. This page displays XRef objects if you are logged in with supervisor rights. You can use the replica list to jump to a real copy of the replica.

Replica Synchronization: Displays the synchronization status of the replica that contains this object.

Entry Synchronization: Shows which attributes need to be synchronized from this server's point of view.

Connection Information: Lets you know where iMonitor got the information for this object.

Entry Information: Displays the names, flags, base class, modification time stamp, and summary of connection information for the object.

Send Entry to all Replicas: Resends this entry's attributes to all other replicas. This process could take some time if the object has many attribute values. This does not make all other copies of the object identical. It simply allows the other replicas to reconsider each attribute.

Viewing Entries for Synchronization or Purging

The Change Cache page lets you view a list of entries that this server needs to consider for synchronization or purging. This option is available only if the server you are accessing is running eDirectory 8.6 or later and the object you are viewing is a partition root. You must have supervisor rights to the NCP server to view this page.

Entry Synchronization: Use this option to determine why an entry needs to be synchronized.

Viewing DirXML Details

Use the DirXML Summary page to view a list of any DirXML drivers running on your server, the status of each driver, any pending associations, and driver details.

1 In iMonitor, click DirXML Summary .

2 Choose from the following options:

Status: Displays the current state of the specified driver. Possible states include: stopped, starting, running, shut down, pending, and getting schema.

Start Option: Displays the current startup option specified for the selected driver.

Pending: Displays the number of associations that have not yet been made.

Driver Details Icon: Displays subscriber and publisher details, XML rules, filters, and pending association lists for DirXML drivers running on your server. Details on the first 50 pending objects are also displayed on this page. The XML rule details provided on this page can be used to determine what to look for in the pending objects to allow their creation to proceed for the specified DirXML driver.

Viewing the Synchronization Status of a Replica

The Replica Synchronization page lets you view the synchronization status of a replica.

1 In iMonitor, click Agent Synchronization in the Assistant frame.

2 Click Replica Synchronization for the partition you want to view.

3 Use the links on this page and in the navigation bar on the left to access other partitions and jump through your replica ring.

Configuring and Viewing Reports

Use the Reports page to view and delete reports run directly on this server. Some reports might take a long time to run and can be resource intensive.

Scheduled reports run without authenticating as a user (that is, as [Public]). Any reports you run directly are run as your identity. All report data is stored on the server from which the report was run.

The Report Config page lets you view a list of preconfigured, custom, and scheduled reports. Use this page to modify and run reports, and to create custom reports for iMonitor pages.

Viewing and Deleting Reports

1 In iMonitor, click Reports .

2 Click to delete a report and to view a report.

Running a Report

1 In iMonitor, click Reports > Report Config.

2 Click to run a report.

Configuring or Scheduling a Report

1 In iMonitor, click Reports > Report Config.

2 Click to configure and schedule a report.

3 Select any options you want.

   3a Click Save Defaults to save the options you selected.

   3b Click Run Report to start the report.

4 (Optional) Configure the report to run on either a periodic basis or at a later time.

   4a Specify a frequency, start time, and start day.

   4b Click Schedule.

Creating a Custom Report

Custom reports let you launch any iMonitor page as a report.

1 In iMonitor, click Reports > Report Config.

2 Click on the Custom Reports line in the Runable Report list.

3 Enter a name for the report > Enter the URL to the iMonitor page you want to launch as a report.

4 Specify the number of versions of the report you want to keep.

5 (Optional) Click Save to save the report.

6 Click Run Report to start the report.

7 (Optional) Configure the report to run on either a periodic basis or at a later time.

   7a Specify a frequency, start time, and start day.

   7b Click Schedule.

Viewing Schema, Class, and Attribute Definitions

Use the Schema page to view your schema, class, and attribute definitions.

1 In iMonitor, click Schema in the Assistant frame.

2 Choose from the following options:

Synchronization List: Lists the servers that this server will synchronize with. This option is available only for servers running NDS eDirectory 8.5 or later. You must have supervisor rights on the server to view this information.

Attribute Definitions: The attribute definition page lists the name of each attribute, the syntax that the attribute value will be in, and the constraints that the attribute operates under. Use the navigation frame on the left to browse for and access individual attributes.

Class Definitions: Lists the name of each class, its rules, and its attributes. Use the navigation frame on the left to browse for and access individual attributes.

Searching for Objects

Use the Search page to search objects based on a variety of query options and filters. The search query options and filters are grouped in two levels of search request forms: basic and advanced. The basic search request form is designed for average users of eDirectory and simple searches. The advanced search request form is designed for advanced users and complicated searches. Currently, only server level search is supported.

All the search options and filters in the four sections are conjunctive. Blank fields (except the Relative Distinguished Name) will be ignored. Use the Ctrl key to deselect an item or select more than one item on the multi-lists. Deselected multi-lists will also be ignored.

1 In iMonitor, click Search 🔍.

2 Click the Help button at the bottom of the search request form to see brief help information added to the form itself.

Click Reload or Refresh to clear the help information.

Using the Stream Viewer

Use the Stream Viewer page to view the current stream in any of the following formats:

- Plain text
- HTML
- GIF
- JPEG
- BMP
- WAV
- Hex Dump
- Other

If you have stream attributes that you consistently want to view in a particular format, you can use the Stream Viewer to choose default display settings.

NDS Stream Attribute Setup: Changes the default display format for streams in your browser. It is up to your browser to display the stream correctly, so it might not always apply the settings you have selected.

You must be authenticated to the server to apply any changes you have made to the default settings. Your changes are stored in STREAMS.INI (for NetWare and Windows NT/2000 servers) or STREAMS.CONF (for Solaris and Linux servers), so you can also manually edit the default settings.

Ensuring Secure iMonitor Operations

iMonitor uses HTTPS for secure iMonitor operations. If the default HTTP port on which iMonitor is listening is 80, the HTTPS port will be 81. If the default HTTP port on which iMonitor is listening is 8008, the HTTPS port is 8009.

For secure ndsimonitor operations on Linux and Solaris systems, you must create a Key Material object (KMO) in the server context. For more information, see "Creating a Key Material Object" on page 79. After creating the KMO, add it to the ndsimonitor configuration file. To do so, add the following line to the /usr/share/ndsimon/ndsimon.conf file, then run the ndsimonitor utility:

```
SSLKey: KMO_name
```

Ensure that there is a space character after the colon character and before the KMO name.

Index Manager

Index Manager is an attribute of the Server object that lets you manage database indexes. These indexes are used by eDirectory to significantly improve query performance.

eDirectory ships with a set of indexes that provide basic query functionality. These default indexes are for the following attributes:

cn	Aliased Object Name
dc	Obituary
Given Name	Member
Surname	Reference
uniqueID	Equivalent to Me
GUID	NLS: Common Certificate

You can also create customized indexes to further improve eDirectory performance in your environment. For example, if your organization has implemented a new LDAP application that looks up an attribute not indexed by default, it might be useful to create an index for that attribute.

NOTE: While indexes improve search performance, additional indexes also add to directory update time. As a general rule, create new indexes only if you suspect performance issues are related to a particular directory lookup.

Using ConsoleOne, you can create or delete indexes. You can also view and manage the properties of an index, including the index name, state, type, rule, and attribute indexed.

Use the Predicate Statistics data to know what additional indexes might be valuable for your environment. See .

Creating an Index

To create an index:

1 In ConsoleOne, right-click the Server object > click Properties > Index Manager > Add.

2 Type the Index Name.

If you do not enter an index name, the attribute is automatically assigned as the index name.

IMPORTANT: The $ character is used as a delimiter for attribute values. If you use the $ character in your index name you must use a preceding backslash (/) character to escape the $ character when working with indexes via LDAP.

3 Select an Attribute.

4 Select the index Rule.

- **Value**: Matches the entire value or the first part of the value of an attribute. For example, value matching could be used to find entries with a "LastName" that is equal to "Jensen," and entries with a "LastName" that begins with "Jen."

- **Presence**: Requires only the presence of an attribute rather than specific attribute values. A query to find all entries with a "Login Script" attribute would use a presence index.

- **Substring**: Matches a subset of the attribute value string. For example, a query to find a "LastName" with "der" would return matches for "Derington," "Anderson," and "Lauder."

   A substring index is the most resource intensive index to create and maintain.

5 Click OK to update the index table.

6 Click Apply to restart limber as a background process and initiate the change.

Deleting an Index

Indexes might outlive their usefulness. You can delete user-defined and auto-created indexes that are no longer a benefit. Use Predicate Stats to help you know which indexes might be less useful.

1 In ConsoleOne, right-click the Server object > click Properties > click Index Manager > click Delete.

2 Select the user or auto added index you want to delete.

3 Click Delete > OK to update the index table.

4 Click Apply to restart limber as a background process and initiate the change.

Taking an Index Offline

During peak times you may want to tune performance by temporarily taking indexes offline. For example, to achieve additional bulk-load speed, you might want to suspend all of the user-defined indexes. Because each object addition or modification requires updating defined indexes, having all indexes active might slow down bulk-loading of data. After the bulk-load is completed, the indexes can be brought online again.

1 In ConsoleOne, right-click the Server object > click Properties > click Index Manager > click Properties.

2 Select the indexes you want to take offline > click Take Offline.

The index state changes from Online to Offline in the display table. An index can be in one of the following states:

 ◆ **Online**: Currently running

 ◆ **Offline**: Suspended; can be started again by clicking Bring Online

 ◆ **New**: Waiting to move to Online

 ◆ **Deleted**: Waiting to be removed from the index table

3 Click OK.

Managing Indexes on Other Servers

If you've found a particular index to be useful on one server, and you see the need for this index on another server, you can copy the index definition from one server to another. In reviewing predicate data, you might also find just the opposite case: an index that was meeting a need for several servers is no longer useful on one of these servers. In that case, you could delete the index from the single server that isn't benefitting from the index.

Index Manager allows you to target a single instance of an index without impacting all instances.

1 In ConsoleOne, right-click the Server object > click Properties > click Index Manager > click Other Servers.

2 To copy an index, select the server you want to copy the index to and choose Create Index.

You can use Load Servers to browse to a server if it is not included in the server list.

If an index with the same attribute and type is found on the destination server, the name and state of that index is changed to the name and state of the index being copied.

3 To delete an index, select the server holding the index you want to delete and choose Delete Index.

4 Click Close.

Using the ndsindex Utility to Manage Indexes on Linux and Solaris Systems

You can use the ndsindex utility to create, list, or delete indexes on Linux or Solaris systems.

Apart from performance reasons, indexes must exist if you want to use the Server Side Sorting (SSS) control or the Virtual List View (VLV) control to perform LDAP searches on the eDirectory LDAP Server. The eDirectory Server holds the indexes maintained by eDirectory. The ndsindex utility helps manage such indexes on Linux or Solaris systems.

- To create an index, use the following syntax:

```
ndsindex add [-h host_name] [-p port] -D user LDAP DN [-W
| -w password] -n eDirectoryServerDN indexDefinition1 [,
indexDefinition2, ....
```

- To list the existing indexes, use the following syntax:

```
ndsindex list [-h host_name] [-p port] -D user LDAP DN [-
W | -w password] -n eDirectoryServerDN [indexname1,
indexname2, ....
```

- To delete indexes, use the following syntax:

```
ndsindex delete [-h host_name] [-p port] -D user LDAP DN
[-W | -w password] -n eDirectoryServerDN indexname1 [,
indexname2, ...
```

Table 48   **ndsindex Options**

Option	Description
-h	Hostname. Specifies the DNS name or IP address of the NDS LDAP Server. The default server is the local host.
-p	Port. Specifies the port number of the NDS LDAP Server. The default port is 389.
-D	user LDAP DN. Specifies the LDAP DN of the user in eDirectory.
-W	Prompts for the password of the LDAP DN specified.
-w	Password. Specifies the password of the LDAP DN in eDirectory.
-n	NDS Server DN. Specifies the NDS Server DN.
indexDefinition	Each indexDefinition holds all the information for one particular index. The string representation is, "<indexName>;<attributeName>;<matchingRule>".
indexName	The name you choose for your index.
attributeName	Name of the eDirectory attribute to be indexed.
matchingRule	matchingRule can be VALUE, SUBSTRING or PRESENCE.
	VALUE: Value matching optimizes queries that involve matching the entire value or the first part of the value specified in the search filter.
	SUBSTRING: Substring matching optimizes queries that involve matching a few characters of the value specified in the search filter.
	PRESENCE: Presence matching optimizes queries that involve only checking for the presence of a specified attribute.

Predicate Data

Predicate data is a server-specific history of the objects people search for. This data and its collection are managed through the ndsPredicateStats object, which is created at the time of eDirectory install. The ndsPredicateStats object name is the server name with a -PS appended.

You can use predicate data to identify most frequently searched for objects and then create indexes to improve the speed of future information access.

Managing Predicate Data

The predicate statistics feature is not intended to be run all the time. Collecting predicate statistics affects search performance. Also, lengthy accumulation of statistics can result in large databases. Use predicate statistics if you suspect performance issues are related to a particular directory look up.

Use the Predicate Data properties page in ConsoleOne to manage the collection of data.

1 In ConsoleOne, right-click the Server object > click Properties > click Predicate Data > click Properties.

2 Specify the appropriate configuration for the ndsPredicateStats object:

Update Interval: Sets the number of seconds to wait before refreshing the data display and writing data to disk.

Advanced > Enable: Specifies whether the collection process should run in the background or should be turned off. If you turn off data collection, the most recently collected data will either be released from memory, or, if you've selected Write to Disk, will be moved to disk.

Advanced > Write to Disk: Determines storage location of predicate data, either always in memory or moving from memory to disk as specified in the Update Interval.

Advanced > Display Value Text: Determines whether the data display will be abbreviated or complete. The abbreviated display provides enough information to determine which predicates are good candidates for indexes.

3 Click OK to update the object configuration.

8 Merging Novell eDirectory Trees

The DSMERGE utility allows you to merge two separate Novell®
eDirectory™ trees into a single eDirectory tree. Only the Tree objects are
merged; container objects and their leaf objects maintain separate identities
within the newly merged Tree.

The two trees you merge are called the local source tree and the target tree. To
merge two trees, you load DSMERGE on a server in the local tree.

DSMERGE does not change Directory names or contexts within the
containers. Object and property rights for the merged objects are retained.

IMPORTANT: You can't merge container or leaf objects with DSMERGE. To move
leaf objects or merge partitions, use ConsoleOne™.

This section contains information on the following:

Using DSMERGE for NetWare

To merge eDirectory trees on NetWare®, use DSMERGE.NLM. DSMERGE
is a loadable module that allows you to merge the root of two separate
eDirectory trees. DSMERGE options let you:

  ◆ Check the status of servers in a tree
  ◆ Check time synchronization
  ◆ Merge two trees
  ◆ Rename a tree
  ◆ Graft a single server tree under a container of another tree

Merging eDirectory Trees on NetWare

The DSMERGE utility allows you to merge two separate eDirectory trees. Only the Tree objects are merged; container objects and their leaf objects maintain separate identities within the newly merged tree.

The two trees you merge are called the source tree and the target tree. The target tree is the tree that the source tree will be merged into. To merge two trees, load DSMERGE on a server in the source tree.

DSMERGE does not change object names within the containers. Object and property rights for the merged tree are retained.

Source Tree Requirements

Novell eDirectory must be installed on the server containing the master replica of the Tree partition. Other servers in the source tree can run any version of NDS$^{®}$ or eDirectory supported by this release.

Target Tree Requirements

Novell eDirectory 8.6 must be installed on the server containing the master replica of the Tree partition. If this server is running any other version of NDS or eDirectory, the merge operation will not complete successfully.

Other servers in the target tree can run any version of NDS or eDirectory supported by this release.

You cannot maintain containers with the same name subordinate to Tree in both the source and target trees. Before merging two trees, one of the containers must be renamed.

If both the source and target trees have a security object, one of them must be removed before merging the trees.

Schema Requirements

Before attempting to perform a merge operation, the schema of both trees must match exactly. You should run DSREPAIR on the server containing the master replica of the Tree partition for each tree. Use the Import Remote Schema option to ensure that each tree is aware of all schema in the other tree.

To access the Import Remote Schema option in DSREPAIR, select the Advanced Options menu > Global Schema Operations > Import Remote Schema. You might have to perform this option on both the source and target tree until no schema differences are reported; otherwise, the merge operation will not succeed.

Merging the Source into the Target Tree

When you merge the trees, the servers in the source tree become part of the target tree.

The target Tree object becomes the new Tree object for objects in the source tree, and the tree name of all servers in the source tree is changed to the target tree's name.

After the merge, the tree name for the target tree servers is retained.

The objects that were subordinate to the source Tree object become subordinate to the target Tree object.

Partition Changes

During the merge, DSMERGE splits the objects below the source Tree object into separate partitions.

All replicas of the Tree partition are then removed from servers in the source tree, except for the master replica. The server that contained the master replica of the source tree receives a replica of the target tree's Tree partition.

Figure 28 on page 239 and Figure 29 on page 240 illustrate the effect on partitions when you merge two trees.

Figure 28   eDirectory Trees before a Merge
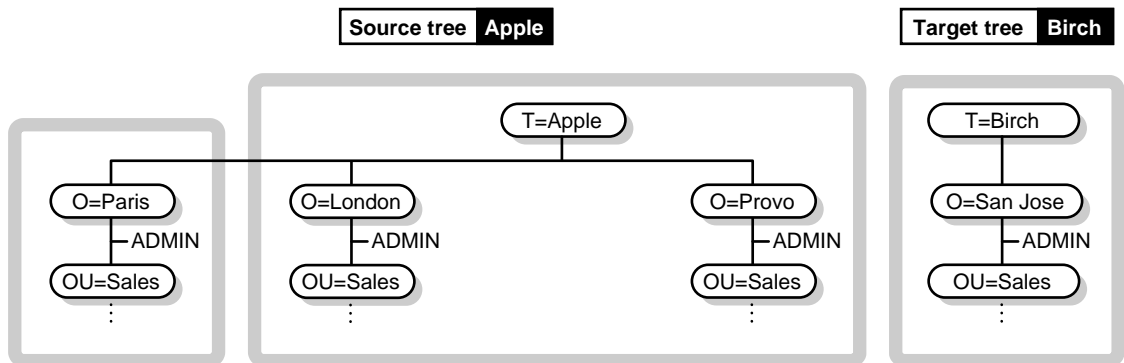
Figure 29   Merged eDirectory Tree
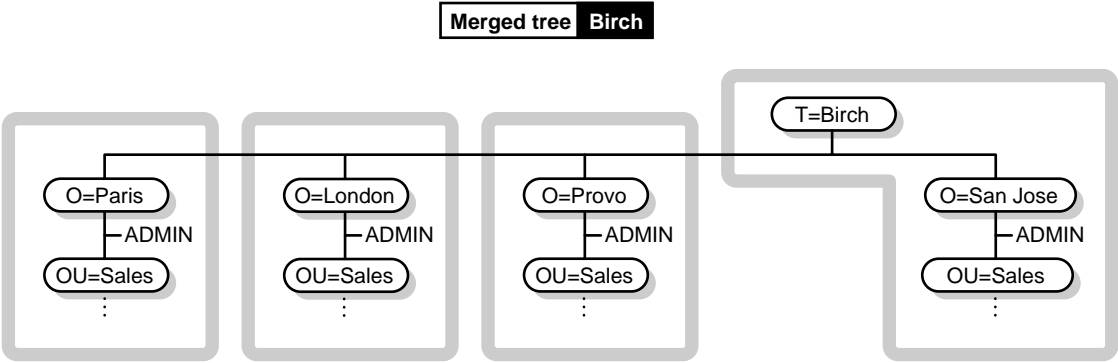
DSMERGE Options

After you load DSMERGE, you can use the following options:

Table 49   DSMERGE Options

Option	Description
Check Servers in This Tree	Contacts all servers (in some cases only a subset of all servers are contacted) in the source tree to verify that each server has the correct version, status, and tree name. The server you are on must have a replica of the Tree partition. It does not require the master replica.
Check Time Synchronization	Displays a list of all servers (in some cases only a subset of all servers are listed) in this tree, along with information about time sources and time synchronization. The server you are on must have a replica of the Tree partition. It does not require the master replica.

Option	Description
Merge Two Trees	Merges the source tree's Tree object to the target tree's Tree object.
	The server you are on must have the master replica of the source tree's Tree partition.
Graft a Single Server Tree	Grafts the root of the source tree under a specified container in the target tree.
Rename Tree	Renames the source tree. Use this option if you are merging two Tree objects with the same name.
	You can rename only the source tree with this option. To rename the target tree, you must load DSMERGE on a server in the target tree and rename it. Then load DSMERGE on the source tree to perform the merge.
	This option requires that the server you are on has the master replica of the partition whose Tree object is the tree name.

Preparing the Source and Target Trees

Before performing a merge operation, ensure that the state of synchronization for all servers affected by the operation is stable. Table 50 provides recommendations for preparing source and target trees for merging.

Table 50   **Recommendations for Preparing Source and Target Trees for Merging**

Prerequisite	Required Action
WANMAN should be turned off on all servers that hold a replica of the source tree's Tree partition or the target tree's Tree partition.	Review your WANMAN policy so that WAN communication restrictions do not interfere with the merge operation. If required, turn WANMAN off before initiating the merge operation.
No aliases or leaf objects can exist at the source tree's Tree object.	Delete any aliases or leaf objects at the source tree's Tree object.

Merging Novell eDirectory Trees   **241**

Prerequisite	Required Action
No similar names can exist between the source and target trees.	Rename objects on the source and target trees if similar names exist.
	Move objects from one of the containers to a different container in its tree if you don't want to rename the container objects, then delete the empty container before running DSMERGE. For more information, see Chapter 4, "Managing Objects," on page 143.
	You can have identical container objects in both trees if they are not immediately subordinate to the Tree object.
No login connections should exist on the source tree.	Close all connections on the source tree.
The eDirectory version must be the same on both the source and target trees.	Upgrade all non-NetWare 5.1 or later servers that have a replica of the root partition.
Any server that contains a replica of the root partition on both the source and target trees must be up and running.	Ensure that all servers containing a replica of the root partition on both source and target trees are up and running.
	Ensure that any WAN links affected are stable.
The schema on both the source and target trees must be the same.	Run DSMERGE. If reports indicate schema problems, use DSREPAIR to match the schemas. (Select Advanced Options Menu > Global Schema Operation > Import Remote Schema to select the tree from which you want to import the schema.) Run DSMERGE again.
Only one tree can have a security container subordinate to the tree root.	If both the source and target tree have the security container, remove one container as explained in Appendix A, "NMAS Considerations," on page 527.

Because the merge operation is one single transaction, it is not subject to catastrophic failure caused by power outages or hardware failure. However, you should perform a regular backup of the eDirectory database before using DSMERGE. For more information, see Chapter 12, "Backing Up and Restoring Novell eDirectory," on page 419.

Synchronizing Time before the Merge

IMPORTANT: Proper configuration of time synchronization is a very involved process. Make sure you allow enough time to synchronize both trees before you merge the trees.

eDirectory will not work properly if different time sources are used that have different times or if all servers in a tree are not time synchronized.

Before you do the merge, make sure that all servers in both trees are time synchronized and that they use only one time server as a time source. However, the target tree time can be ahead of the source tree time by as much as five minutes.

Generally, there should be only one Reference or one Single time server in a tree. Likewise, after the merge, the tree should contain only one Reference or one Single time server. For more information on time server types, refer to *Network Time Management Administration Guide* in the NetWare 6 documentation set on the Novell Documentation Web site (http://www.novell.com/documentation).

If each of the trees you are merging has either a Reference or a Single time server, reassign one of them to refer to the Reference or Single time server in the other tree so that the final tree contains only one Reference or Single time server.

To view time synchronization information, see "Checking Time Synchronization" on page 246.

Checking Servers in the Tree

Before you rename or merge trees, use the Check Servers in This Tree option to contact all servers in the tree and verify that all servers have the same tree name.

After you rename or merge trees, use this option to verify that all servers have the new tree name.

From the server console:

1 On the server where a replica of the root partition of the source tree is stored, type **dsmerge**.

2 Select Check Servers in This Tree.

Each server in the tree is listed in the Status of Servers in the Tree screen, with its corresponding status information. Any servers that have existing problems are flagged and then listed at the top of the server list.

You should confirm that each server's status is marked as Verified before completing a merge.

Table 51 describes the information provided in the Status of Servers in the Tree screen.

Table 51   Status of Servers in the Tree Fields

Field	Operation
Server Name	Lists the names of all servers DSMERGE contacted, and shows their context within the tree.
Version	Indicates the version of NetWare running on the server.

Field	Operation
Status	◆ Up Indicates that the server is in the right tree. ◆ Error *number* All eDirectory errors are numbered between –600 and –799 in decimal notation. For information about a specific error code, search the Error Codes. ◆ Unknown Indicates that the server is not responding. This usually indicates a downed server or communication problems. ◆ Wrong Tree Indicates that this server does not belong to this eDirectory tree. This status might occur if the tree was recently merged or renamed because the server might take a few minutes to recognize the change. This status can also occur if the server was reinstalled in another tree, but not properly removed from this tree. If so, delete this server's object from this tree.

Checking Time Synchronization

Use this procedure on both trees before merging them.

1 From the server console of the server that contains a master replica of the Tree partition of the source tree, type **DSMERGE**.

If you don't know where the master replica is, load DSMERGE on any server in the source tree. You will be prompted with the name of the server that contains the master replica when it is required.

2 Select Check Time Synchronization.

The Time Synchronization Information for the Tree *tree_name* screen appears.

This option lists all servers in the tree, along with information about their time sources and the server time.

Verify that all servers in the tree are synchronized and that they are using the same time source.

Table 52 describes the information provided in the Time Synchronization Information for the Tree *tree_name* screen.

Table 52   Time Synchronization Information for the Tree Options

Field	Operation
Server Name	Lists the name of each server.
Type	Indicates one of the following time server types: Reference, Single, Primary, and Secondary. If the server could not be contacted, it is listed as Unknown. In general, there should be only one Reference or Single reference (not both) in a tree.

Field	Operation
In Sync	◆ Yes Indicates that the server is in sync with a time server. You need to manually check that all servers are using the same time source. ◆ No Indicates that the server is not in sync with a time server. This option does not determine if the source server is in sync with the server you have selected. It only reports whether it is currently in a synchronized state. If the server has temporarily lost synchronization with its time source, it might still have the correct time. Check the type of time server each server is using as a time source to determine if they are using different time servers.
Time Delta	Displays the difference in time between the source server and the selected server in the list. If the difference in time is more than a few seconds, it might indicate that the servers are using different time sources.

Merging Two Trees

For complete functionality of all menu options, run DSMERGE on a server that contains the master replica of the Tree partition.

If you don't know where the master replica is stored, you will be prompted with the correct server name when you attempt an operation that requires the master replica.

To perform a merge operation, you must load DSMERGE on the source tree.

When merging large trees, it's significantly faster to designate the source tree as the tree with fewer objects immediately subordinate to the Tree object. By doing this, you create fewer partition splits during the merge, since all objects subordinate to the Tree object result in new partitions.

Because the source tree name no longer exists after the merge, you might need to change your client workstation configurations. For Novell Client for DOS/Windows, check the Preferred Tree and Preferred Server statements in the NET.CFG files. For Novell Client for Windows NT/2000 and Windows 95/98, check the Preferred Tree and Preferred Server statements on the client Property Page.

If Preferred Server is used, the client is unaffected by a tree merge or rename operation because the client still logs in to the server by name. If Preferred Tree is used and the tree is renamed or merged, then that tree name no longer exists. Only the target tree name is retained after the merge. Change the preferred tree name to the new tree name.

To minimize the number of client workstations you need to update, designate the tree with the most client workstations as the target tree, because the final tree retains the name of the target tree.

Or rename the tree after the merge operation so that the final tree name corresponds to the tree with the greater number of client workstations attaching to it. For more information, see . Plan on a period of down time to allow for the tree merge and then a tree rename.

Use the following list of prerequisites to determine readiness for the merge operation:

❑ Access to the server console on the source tree or an established RCONSOLE session with that server.

❑ The name and password of the Administrator objects that have Supervisor object rights to the Tree object of both trees you want to merge.

❑ A backup of the eDirectory database for the two trees.

❑ All servers in both trees are synchronized and using the same time source.

❑ (Optional) All servers in the tree are operational. (Servers that are down will update automatically when they are operational.)

The merge process itself only takes a few minutes but there are other variables that increase the length of time for the merge operation to complete. These factors are as follows:

- If there are a lot of objects subordinate to the Tree object that must be split into partitions.

- If there are a lot of servers in the source tree that require a tree name change.

To merge two trees:

1 On the server that stores the master replica on the source tree, type **DSMERGE**.

If you don't know where the master replica is stored, you will be prompted with the correct server name when you attempt to merge the trees.

2 Select Merge Two Trees.

3 Enter the administrator name and password to log in to the source tree.

Log in as a user who has the Supervisor object right to the Tree object on the source tree. Enter the typeless or typeful distinguished name, such as **admin.novell** or **cn=admin.o=novell**. Entering only **admin** is invalid because it is not the complete name of the User object.

4 Select Target Tree > select a target tree from the list of servers in the Available Trees window.

If the tree you want is not in the list, press Insert > enter the target tree's network address.

5 Enter the administrator name and password to log in to the target tree.

6 Press F10 to perform the merge.

A message stating that the trees have been merged successfully is displayed.

Renaming the Tree

You must rename a tree if the two trees you want to merge have the same name.

You can rename only the source tree. To rename the target tree, run DSMERGE from a server on the target tree.

If you change the tree name, the bindery context does not automatically change also, including the one set in the AUTOEXEC.NCF file. Because the bindery context set in the AUTOEXEC.NCF file also contains the tree name (for example, SET Bindery Context = O=n.<test_tree_name>), the server with the recently changed tree name does not use the context that it used before the tree name change.

After you change a tree's name, you might need to change your client workstation configurations. For Novell Client for DOS/Windows, check the Preferred Tree and Preferred Server statements in the NET.CFG files. For Novell Client for Windows NT/2000 and Windows 95/98, check the Preferred Tree and Preferred Server statements on the client Property Page.

If Preferred Server is used, the client is unaffected by a tree merge or rename operation because the client still logs in to the server by name. If Preferred Tree is used and the tree is renamed or merged, then that tree name no longer exists. Only the target tree name is retained after the merge. Change the preferred tree name to the new tree name.

When you merge two trees, to minimize the number of client workstations that need to be updated, designate the tree with the most client workstations as the target tree because the final tree retains the name of the target tree.

Or rename the tree after the merge so that the final tree name corresponds to the tree name with the majority of client workstations.

Another option is to rename the merged tree to the name of the original source tree. If you choose this option, then you must update the NET.CFG files on the target tree client workstations.

Use the following list of prerequisites to determine readiness for the renaming operation:

❑ Access to a server console on the source tree or an established RCONSOLE session with the server.

❑ The Supervisor object right to the Tree object of the source tree.

❑ (Optional) All servers in the tree are operational. (Servers that are down will update automatically when they are operational.)

To rename the tree:

1 On the server that stores a master replica of the partition whose Tree object is the tree name, type **DSMERGE**.

   If you don't know where the master replica is, load DSMERGE on any server in the source tree. Then you will be prompted with the correct server name when you attempt to rename a tree.

2 Select Rename This Tree.

3 Enter the administrator name and password to log in to the source tree.

Log in as a user who has the Supervisor object right to the Tree object on the source tree. Enter your complete name, such as **admin.novell** or **cn=admin.o=novell**. Entering only **admin** is invalid since it is not a complete name.

4 Enter the new tree name.

5 Press F10 to perform the rename.

Completing the Tree Merge

Following the merging of two trees, it might be necessary to complete the following tasks:

1 (Optional) Select Checking Servers in the Tree in the DSMERGE main menu to confirm that all tree names were changed correctly.

For more information, see "Checking Servers in the Tree" on page 244.

2 Check the new partitions that the merge operation created.

If you have many small partitions in the new tree, or if you have partitions that contain related information, you might want to merge them. For more information, see "Merging a Partition" on page 163.

3 Copy a new replica to any non-NetWare 5 servers after the merge is complete, if you did not upgrade before running DSMERGE.

4 Re-create any leaf objects or aliases in the tree that were deleted before you ran DSMERGE.

5 Evaluate partitioning of the eDirectory tree.

Merging trees might change replica placement requirements on the new tree. You should carefully evaluate and change the partitioning as needed.

6 Update your client workstation configuration.

For Novell Client for DOS/Windows, check the Preferred Tree and Preferred Server statements in the NET.CFG files. For Novell Client for Windows NT/2000 and Windows 95/98, check the Preferred Tree and Preferred Server statements on the client Property Page, or rename the target tree.

If Preferred Server is used, the client is unaffected by a tree merge or rename operation because the client still logs in to the server by name. If Preferred Tree is used and the tree is renamed or merged, then that tree name no longer exists. Only the target tree name is retained after the merge. Change the preferred tree name to the new tree name.

HINT: To minimize the number of NET.CFG files you need to update, designate the tree with the most client workstations as the target tree because the final tree retains the name of the target tree. Or rename the tree after the merge operation so that the final tree name corresponds to the majority of the client workstations' NET.CFG files. For more information, see "Renaming the Tree" on page 249.

The Access Control List (ACL) for the Tree object of the source tree is preserved. Therefore, the rights of the source tree's user Admin to the Tree object are still valid.

After the merge is complete, both admin users still exist and are uniquely identified by different container objects.

For security reasons, you might want to delete one of the two Admin User objects or restrict the rights of the two objects.

Grafting a Single Server Tree

Understanding Grafting

The Graft Tree option lets you graft a single-server source tree's Tree object under a container specified in the target tree. After the graft is completed, the source tree receives the target tree's name.

During the graft, DSMERGE changes the object class of the source tree's Tree object to Domain and makes a new partition. The new Domain object is the partition root for the new partition. All the objects under the source tree's Tree object are located under the Domain object.

The target tree's administrator has rights to the resulting tree's root container and therefore has rights to the source tree's grafted root.

NOTE: It may take up to several hours for the inherited rights to be recalculated and become effective. This time will vary based on the tree's complexity, size, and number of partitions.

The source tree's administrator has rights only in the newly created domain object.

Figure 30 on page 253 and Figure 31 on page 254 illustrate the effect when you graft a tree into a specific container.

Figure 30   eDirectory Trees before a Graft

Source tree Preconfigured_tree

T=Preconfigured_tree

OU=Cache Services   OU=GroupWise   OU=IS
                                                      ADMIN

Target tree Oak

T=Oak_tree

O=San Jose   🔒 Security

ADMIN

OU=Engineering   OU=Operations

OU=New Devices
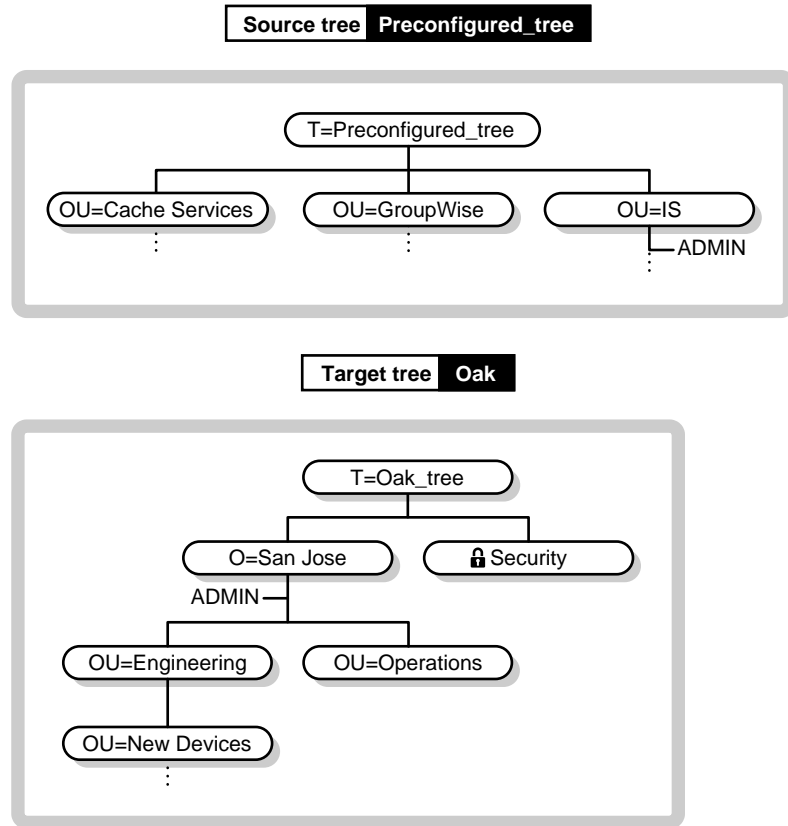
Figure 31   eDirectory Tree after a Graft

Target tree   Oak

Source tree grafted under the New Devices container

```
                        ┌──────────────┐
                        │  T=Oak_tree  │
                        └──────────────┘
              ┌────────────────┴──────────────────┐
       ┌─────────────┐                    ┌──────────────┐
       │ O=San Jose  │                    │ 🔒 Security  │
       └─────────────┘                    └──────────────┘
     ADMIN ──┤
        ┌──────────────────┐      ┌──────────────────┐
        │ OU=Engineering   │      │ OU=Operations    │
        └──────────────────┘      └──────────────────┘
     ┌──────────────────┐
     │ OU=New Devices    │
     └──────────────────┘

                ┌────────────────────────┐
                │ DC=Preconfigured_tree  │
                └────────────────────────┘
     ┌────────────────────┬──────────────────┬──────────────┐
┌──────────────────┐  ┌──────────────────┐  ┌──────────────┐
│ OU=Cache Services│  │ OU=GroupWise     │  │ OU=IS        │
└──────────────────┘  └──────────────────┘  └──────────────┘
      ⋮                     ⋮                   └─ ADMIN
                                                   ⋮
```

Context Name Changes

After the graft of the source tree into the target tree container, the distinguished names for objects in the source tree will be appended with the source tree's name followed by the distinguished name of the target tree's container name where the source tree was merged. The relative distinguished name will remain the same.

For example, if you are using dot delimiters, the typeful name for Admin in the Preconfigured_tree (source tree) is:

```
CN=Admin.OU=IS.T=Preconfigured_tree
```

After the Preconfigured_tree is merged into the New Devices container in the Oak_tree, the typeful name for Admin is:

```
CN=Admin.OU=IS.DC=Preconfigured_tree.OU=Newdevices.
  OU=Engineering.O=Sanjose.T=Oak_tree.
```

NOTE: The distinguished name maximum character length is 256 characters. This limitation is particularly important when you are grafting the root of one tree into a container near the bottom of the target tree.

The last dot following Oak_tree (Oak_tree.) indicates that the last element in the distinguished name is the tree name. If you leave off the trailing dot, then leave off the tree name.

Preparing the Source and Target Trees

Before initiating the graft operation, ensure that the state of all of the servers affected by the operation is stable. Table 53 provides recommendations for preparing the source and target trees before grafting.

Table 53   Recommendations for Preparing Source and Target Trees before Grafting

Prerequisite	Required Action
WANMAN should be turned off on all servers that hold a replica of the source tree's Tree partition or the target tree's Tree partition.	Review your WANMAN policy so that WAN communication restrictions do not interfere with the merge operation. If required, turn WANMAN off before initiating the merge operation.
The source tree must have only one server.	Remove all but one server from the source tree.
No aliases or leaf objects can exist at the source tree's Tree object.	Delete any aliases or leaf objects at the source tree's Tree object.
No similar names can exist in the graft container.	Rename objects under the target tree graft container or rename the source tree.
	Move objects from one of the containers to a different container in its tree if you don't want to rename objects, then delete the empty container before running DSMERGE. For more information, see Chapter 4, "Managing Objects," on page 143.
	You can have identical container objects in both trees if they are not immediately subordinate to the same parent object. Objects are uniquely identified by their immediate container object.

Prerequisite	Required Action
The eDirectory version for both the source tree and target tree container must be 8.51 SP2a or later.	DSMERGE will search for the appropriate version of eDirectory. If an acceptable version isn't found, DSMERGE will return an error. You can get the latest version of eDirectory from the product download page (http://www.novell.com/download/).
The container where you will join the target tree is in a partition that has no replicas (a single-server partition).	If the target container has multiple replicas: • Make the partition associated with this container the master replica, then delete other replicas, or • Split the target tree graft container in to a separate partition and then remove replicas. After the graft is complete, the partition association can be re-established.
The server holding the target container must also hold a replica of the ROOT partition.	If the server doesn't hold a replica of ROOT, the graft will fail, and you will see error -672 No Access because the directory will be unable to verify administrator rights for the target tree. Use ConsoleOne to add a replica for ROOT. For more information, see Adding a Replica (http://www.novell.com/documentation/lg/consol13/c1_enu/data/a3j28dn.html#a3zztpi) in the ConsoleOne User Guide (http://www.novell.com/documentation/lg/consol13/index.html).

Prerequisite	Required Action
The schema on both the source and target trees must be the same.	Run the Graft option in DSMERGE. If reports indicate schema problems, run DSREPAIR on the target tree to import the schema from the source tree.
	The graft operation automatically imports the schema from the target tree to the source tree.
	Run DSMERGE again.
Only one tree can have a security container subordinate to the tree root.	If both the source and target tree have the security container, remove one container as explained in Appendix A, "NMAS Considerations," on page 527.
The source tree's time reference must be reconfigured.	Generally the source tree should be set as a secondary server configured to get its time source from a server in the target tree.
	To reconfigure Timesync, see Configuring Timesync on Servers in the Network Time Management Administration Guide.

Containment Requirements for Grafting

To graft a source tree into a target tree container requires that the target tree container be prepared to accept the source tree. The target tree container must be able to contain an object of the class domain. If there is a problem with containment, error -611 Illegal Containment will occur during the graft operation.

Use the information in Table 54 to determine if you need to run DSREPAIR to modify containment lists.

Table 54   **Target and Source Tree Container Requirements**

Target Tree Container Requirements	The target tree container object must include the domain object in its containment list.
	You can check this using iMonitor > Schema. If the containment list does not include Domain, run DSREPAIR to make schema enhancements.
Source Tree Requirements	The graft operation changes the source tree root from the class Tree Root to the class Domain. All of the object classes that are subordinate to the Tree must be able to be contained by the class Domain according to the schema rules.
	You can check this using iMonitor > Schema. If the containment list does not include Domain, run DSREPAIR to make schema enhancements.

If containment requirements aren't met, run DSREPAIR to correct the schema.

1 Load DSPEPAIR, select Advanced Menu > Global Schema Operations.

2 Select Optional Schema Enhancements.

    The schema is updated and results are displayed on screen.

Grafting the Source and Target Tree

After you ensure that prerequisites are met, use DSMERGE to perform the graft as explained in the following procedure:

1 Load DSMERGE at the console of the *source* tree server > select Graft a Single Server Tree.

2 Enter the required data including user name and password for both the source and target tree and the distinguished name of the target tree container, for example,
`New Devices.Engineering.San Jose.Oak_tree`

3 Press F10 to start the graft.

Using DSMERGE for Windows NT/2000

To merge eDirectory trees on Windows NT/2000, use DSMERGE.DML. DSMERGE is a loadable module that allows you to merge the root of two separate eDirectory trees. DSMERGE options let you:

- Check the status of servers in a tree
- Check time synchronization
- Merge two trees
- Rename a tree
- Graft a single server tree under a container of another tree

Merging eDirectory Trees on Windows NT/2000

The DSMERGE utility allows you to merge two separate eDirectory trees. Only the Tree objects are merged; container objects and their leaf objects maintain separate identities within the newly merged tree.

The two trees you merge are called the source tree and the target tree. The target tree is the tree that the source tree will be merged into. To merge two trees, load DSMERGE on a server in the source tree.

DSMERGE does not change object names within the containers. Object and property rights for the merged tree are retained.

Source Tree Requirements

Novell eDirectory must be installed on the server containing the master replica of the Tree partition. Other servers in the source tree can run any version of NDS or eDirectory supported by this release. See "Installing eDirectory for Windows NT/2000 Server" on page 26 for more information.

Target Tree Requirements

Novell eDirectory 8.6 must be installed on the server containing the master replica of the Tree partition. If this server is running any other version of NDS or eDirectory, the merge operation will not complete successfully.

Other servers in the target tree can run any version of NDS or eDirectory supported by this release.

You cannot maintain containers with the same name subordinate to Tree in both the source and target trees. Before merging two trees, one of the containers must be renamed.

If both the source and target trees have a security object, one of them must be removed before merging the trees.

Schema Requirements

Before attempting to perform a merge operation, the schema of both trees must match exactly. You should run DSREPAIR on the server containing the master replica of the Tree partition for each tree. Use the Import Remote schema option to ensure that each tree is aware of all schema in the other tree.

To access the Import Remote Schema option in DSREPAIR on Windows NT/2000:

1 Start NDSCONSOLE by running NDSCONS.EXE.

2 Select DSREPAIR.DML > click Start.

3 Click Schema > Import Remote Schema.

You might have to perform this option on both the source and target tree until no schema differences are reported; otherwise, the merge operation will not succeed.

Merging the Source into the Target Tree on Windows NT/2000

When you merge the trees, the servers in the source tree become part of the target tree.

The target Tree object becomes the new Tree object for objects in the source tree, and the tree name of all servers in the source tree is changed to the target tree's name.

After the merge, the tree name for the target tree servers is retained.

The objects that were subordinate to the source Tree object become subordinate to the target Tree object.

Partition Changes

During the merge, DSMERGE splits the objects below the source Tree object into separate partitions.

All replicas of the Tree partition are then removed from servers in the source tree, except for the master replica. The server that contained the master replica of the source tree receives a replica of the target tree's Tree partition.

Figure 32 and Figure 33 on page 262 illustrate the effect on partitions when you merge two trees.

Figure 32   eDirectory Trees before a Merge
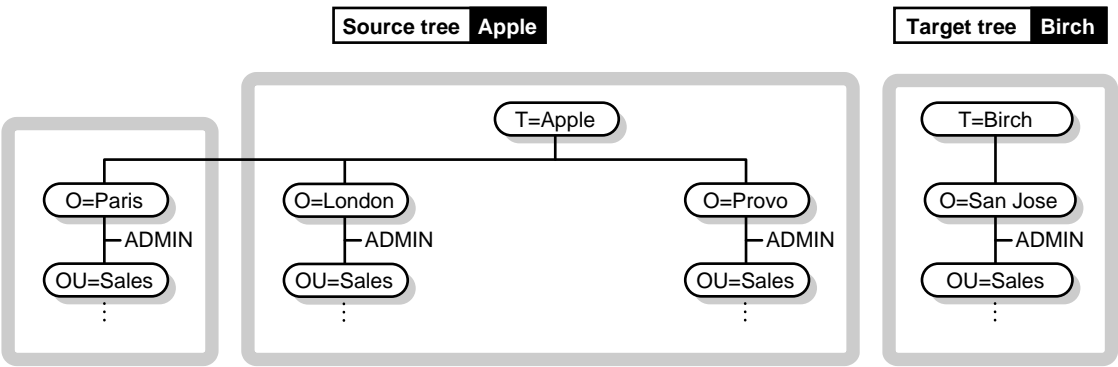
Figure 33   Merged eDirectory Tree
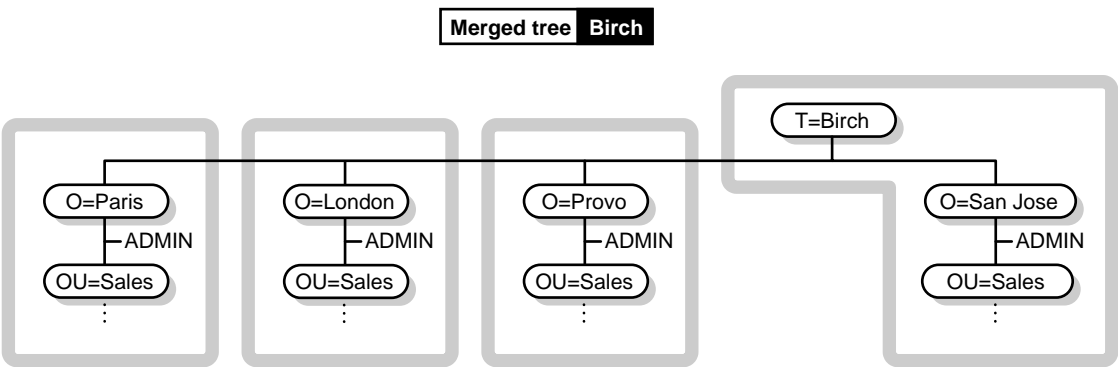
DSMERGE Options

After you load DSMERGE, you can use the following options:

Table 55   **DSMERGE Options**

Option	Description
Check Servers in This Tree	Contacts all servers (in some cases only a subset of all servers are contacted) in the source tree to verify that each server has the correct version, status, and tree name.
	The server you are on must have a replica of the Tree partition. It does not require the master replica.
Check Time Synchronization	Displays a list of all servers (in some cases only a subset of all servers are listed) in this tree, along with information about time sources and time synchronization.
	The server you are on must have a replica of the Tree partition. It does not require the master replica.
Merge Two Trees	Merges the source tree's Tree object to the target tree's Tree object.
	The server you are on must have the master replica of the source tree's Tree partition.
Graft a Single Server Tree	Grafts the root of the source tree under a specified container in the target tree.

Option	Description
Rename Tree	Renames the source tree. Use this option if you are merging two Trees with the same name.
	You can rename only the source tree with this option. To rename the target tree, you must load DSMERGE on a server in the target tree and rename it. Then load DSMERGE on the source tree to perform the merge.
	This option requires that the server you are on has the master replica of the partition whose Tree object is the tree name.

Preparing the Source and Target Trees

Before performing a merge operation, ensure that the state of synchronization for all servers affected by the operation is stable. Table 56 provides recommendations for preparing source and target trees for merging.

Table 56   **Recommendations for Preparing Source and Target Trees for Merging**

Prerequisite	Required Action
WANMAN should be turned off on all servers that hold a replica of the source tree's Tree partition or the target tree's Tree partition.	Review your WANMAN policy so that WAN communication restrictions do not interfere with the merge operation. If required, turn WANMAN off before initiating the merge operation.
No aliases or leaf objects can exist at the source tree's Tree object.	Delete any aliases or leaf objects at the source tree's Tree object.

Prerequisite	Required Action
No similar names can exist between the source and target trees.	Rename objects on the source and target trees if similar names exist.
	Move objects from one of the containers to a different container in its tree if you don't want to rename the container objects, then delete the empty container before running DSMERGE. For more information, see Chapter 4, "Managing Objects," on page 143.
	You can have identical container objects in both trees if they are not immediately subordinate to the Tree.
No login connections should exist on the source tree.	Close all connections on the source tree.
The eDirectory version must be the same on both the source and target trees.	Upgrade all non-NetWare 5.1 or later servers that have a replica of the root partition.
Any server that contains a replica of the root partition on both the source and target trees must be up and running.	Ensure that all servers containing a replica of the root partition on both source and target trees are up and running.
	Ensure that any WAN links affected are stable.
The schema on both the source and target trees must be the same.	Run DSMERGE. If reports indicate schema problems, use DSREPAIR to match the schemas. (Select Advanced Options Menu > Global Schema Operation > Import Remote Schema to select the tree from which you want to import the schema.) Run DSMERGE again.
Only one tree can have a security container subordinate to the tree root.	If both the source and target tree have the security container, remove one container as explained in Appendix A, "NMAS Considerations," on page 527.

Because the merge operation is one single transaction, it is not subject to catastrophic failure caused by power outages or hardware failure. However, you should perform a regular backup of the eDirectory database before using DSMERGE. For more information, see Chapter 12, "Backing Up and Restoring Novell eDirectory," on page 419.

Synchronizing Time before the Merge

IMPORTANT: Proper configuration of time synchronization is a very involved process. Make sure you allow enough time to synchronize both trees before you merge the trees.

eDirectory will not work properly if different time sources are used that have different times or if all servers in a tree are not time synchronized.

Before you do the merge, make sure that all servers in both trees are time synchronized and that they use only one time server as a time source. However, the target tree time can be ahead of the source tree time by as much as five minutes.

Generally, there should be only one Reference or one Single time server in a tree. Likewise, after the merge, the tree should contain only one Reference or one Single time server. For more information on time server types, refer to *Network Time Management Administration Guide* in the NetWare 6 documentation set on the Novell Documentation Web site (http://www.novell.com/documentation).

If each of the trees you are merging has either a Reference or a Single time server, reassign one of them to refer to the Reference or Single time server in the other tree so that the final tree contains only one Reference or Single time server.

To view time synchronization information, see "Checking Time Synchronization" on page 246.

Checking Servers in the Tree

Before you rename or merge trees, verify that all servers have the same tree name.

After you rename or merge trees, verify that all servers have the new tree name.

1 Start NDSCONSOLE by running NDSCONS.EXE.

2 Select DSMERGE.DLM > click Start.

If you don't know where the master replica is stored, you will be prompted with the correct server name when you attempt to merge the trees.

3 From the NDS Merge screen, verify the server name, DS version, and status for each server in the tree.

Each server in the tree is listed in the NDS Merge screen, with its corresponding status information. Any servers that have existing problems are flagged and then listed at the top of the server list.

You should confirm that each server's status is marked as Verified before completing a merge.

Table 57 describes the information provided in the NDS Merge screen.

Table 57   NDS Merge Screen Options

Field	Operation
Server Name	Lists the names of all servers DSMERGE contacted, and shows their context within the tree.
DS Version	Indicates the version of eDirectory running on the server.

Field	Operation
Status	◆ Up
	Indicates that the server is in the right tree.
	◆ Error *number*
	All eDirectory errors are numbered between –600 and –799 in decimal notation.
	For information about a specific error code, search the Error Codes.
	◆ Unknown
	Indicates that the server is not responding. This usually indicates a downed server or of communication problems.
	◆ Wrong Tree
	Indicates that this server does not belong to this directory tree.
	This status might occur if the tree was recently merged or renamed because the server might take a few minutes to recognize the change.
	This status can also occur if the server was reinstalled in another tree, but not properly removed from this tree. If so, delete this server's object from this tree.

Checking Time Synchronization

Use this procedure on both trees before merging them.

1 Start NDSCONSOLE by running NDSCONS.EXE.

2 Select DSMERGE.DLM > click Start.

If you don't know where the master replica is stored, you will be prompted with the correct server name when you attempt to merge the trees.

3 From the NDS Merge screen, verify that all servers in the tree are synchronized and that they are using the same time source.

Table 58 describes the information provided in the Time Synchronization Information for the Tree *tree_name* screen.

Table 58   **Time Synchronization Information for the Tree Options**

Field	Operation
Server Name	Lists the name of each server.
Type	Indicates one of the following time server types: Reference, Single, Primary, and Secondary. If the server could not be contacted, it is listed as Unknown. In general, there should be only one Reference or Single reference (not both) in a tree.
In Sync	◆ Yes Indicates that the server is in sync with a time server. You need to manually check that all servers are using the same time source. ◆ No Indicates that the server is not in sync with a time server. This option does not determine if the source server is in sync with the server you have selected. It only reports whether it is currently in a synchronized state. If the server has temporarily lost synchronization with its time source, it might still have the correct time. Check the type of time server each server is using as a time source to determine if they are using different time servers.

Field	Operation
Time Delta	Displays the difference in time between the source server and the selected server in the list.
	If the difference in time is more than a few seconds, it might indicate that the servers are using different time sources.

Merging Two Trees

For complete functionality of all menu options, run DSMERGE on a server that contains the master replica of the Tree partition.

If you don't know where the master replica is stored, you will be prompted with the correct server name when you attempt an operation that requires the master replica.

To perform a merge operation, you must load DSMERGE on the source tree.

When merging large trees, it's significantly faster to designate the source tree as the tree with fewer objects immediately subordinate to the Tree. By doing this, you create fewer partition splits during the merge, since all objects subordinate to the Tree object result in new partitions.

Because the source tree name no longer exists after the merge, you might need to change your client workstation configurations. For Novell Client™ for DOS/Windows, check the Preferred Tree and Preferred Server statements in the NET.CFG files. For Novell Client for Windows NT/2000 and Windows 95/98, check the Preferred Tree and Preferred Server statements on the client Property Page.

If Preferred Server is used, the client is unaffected by a tree merge or rename operation because the client still logs in to the server by name. If Preferred Tree is used and the tree is renamed or merged, then that tree name no longer exists. Only the target tree name is retained after the merge. Change the preferred tree name to the new tree name.

To minimize the number of client workstations you need to update, designate the tree with the most client workstations as the target tree, because the final tree retains the name of the target tree.

Or rename the tree after the merge operation so that the final tree name corresponds to the tree with the greater number of client workstations attaching to it. For more information see "Renaming the Tree" on page 272. Plan on a period of down time to allow for the tree merge and then a tree rename.

Use the following list of prerequisites to determine readiness for the merge operation:

❑ Access to the server console on the source tree.

❑ The name and password of the administrator objects that have Supervisor object rights to the Tree object of both trees you want to merge.

❑ A backup of the eDirectory database for the two trees.

❑ (Optional) All servers in the tree are operational. (Servers that are down will update automatically when they are operational.)

The merge process itself only takes a few minutes but there are other variables that increase the length of time for the merge operation to complete. These factors are as follows:

◆ If there are a lot of objects subordinate to the Tree object that must be split into partitions.

◆ If there are a lot of servers in the source tree that require a tree name change.

To merge two trees:

1 Start NDSCONSOLE by running NDSCONS.EXE.

2 Select DSMERGE.DLM > click Start.

   If you don't know where the master replica is stored, you will be prompted with the correct server name when you attempt to merge the trees.

3 Click Operations > Merge Two Trees.

4 Enter the administrator name and password to log in to the source tree.

   Log in as a user who has the Supervisor object right to the Tree object on the source tree. Enter the typeless or typeful distinguished name, such as `admin.novell` or `cn=admin.o=novell`. Entering only `admin` is invalid because it is not the complete name of the User object.

5 From the Target Name drop-down list, select a target tree from the list of servers in the Available Trees window.

If the tree you want is not in the list, enter the target tree's IP address.

6 Enter the administrator name and password to log in to the target tree.

7 Click OK to perform the merge.

A message stating that the trees have been merged successfully is displayed.

Renaming the Tree

You must rename a tree if the two trees you want to merge have the same name.

You can rename only the source tree. To rename the target tree, run DSMERGE from a server on the target tree.

After you change a tree's name, you might need to change your client workstation configurations. For Novell Client for DOS/Windows, check the Preferred Tree and Preferred Server statements in the NET.CFG files. For Novell Client for Windows NT/2000 and Windows 95/98, check the Preferred Tree and Preferred Server statements on the client Property Page.

If Preferred Server is used, the client is unaffected by a tree merge or rename operation because the client still logs in to the server by name. If Preferred Tree is used and the tree is renamed or merged, then that tree name no longer exists. Only the target tree name is retained after the merge. Change the preferred tree name to the new tree name.

When you merge two trees, to minimize the number of client workstations that need to be updated, designate the tree with the most client workstations as the target tree because the final tree retains the name of the target tree.

Or rename the tree after the merge so that the final tree name corresponds to the tree name with the majority of client workstations.

Another option is to rename the merged tree to the name of the original source tree. If you select this option, then you must update the NET.CFG files on the target tree client workstations.

Use the following list of prerequisites to determine readiness for the renaming operation:

❑ Access to a server console on the source tree.

❑ The Supervisor object right to the Tree object of the source tree.

❑ (Optional) All servers in the tree are operational. (Servers that are down will update automatically when they are operational.)

To rename the tree:

1 On the server that stores a master replica of the partition whose Tree object is the tree name, start NDSCONSOLE by running NDSCONS.EXE.

2 Select DSMERGE.DLM > click Start.

If you don't know where the master replica is, load DSMERGE on any server in the source tree. Then you will be prompted with the correct server name when you attempt to rename a tree.

3 Click Operations > Rename This Tree.

4 Enter the administrator name and password to log in to the source tree.

Log in as a user who has the Supervisor object right to the Tree object on the source tree. Enter your complete name, such as `admin.novell` or `cn=admin.o=novell`. Entering only `admin` is invalid since it is not a complete name.

5 Enter the new tree name.

6 Click OK to perform the rename.

Completing the Tree Merge

Following the merging of two trees, it might be necessary to complete the following tasks:

1 From the NDS Merge screen, verify that all tree names were changed correctly.

For more information, see "Checking Servers in the Tree" on page 244.

2 Check the new partitions that the merge operation created. If you have many small partitions in the new tree, or if you have partitions that contain related information, you might want to merge them.

For more information, see "Merging a Partition" on page 163.

3 Copy a new replica to any non-NetWare 5 servers after the merge is complete, if you did not upgrade before running DSMERGE.

4 Re-create any leaf objects or aliases in the Tree that were deleted before you ran DSMERGE.

5 Evaluate partitioning of the eDirectory tree.

Merging trees might change replica placement requirements on the new tree. You should carefully evaluate and change the partitioning as needed.

6 Update your client workstation configuration.

For Novell Client for DOS/Windows, check the Preferred Tree and Preferred Server statements in the NET.CFG files. For Novell Client for Windows NT/2000 and Windows 95/98, check the Preferred Tree and Preferred Server statements on the client Property Page. Or rename the target tree.

If Preferred Server is used, the client is unaffected by a tree merge or rename operation because the client still logs in to the server by name. If Preferred Tree is used and the tree is renamed or merged, then that tree name no longer exists. Only the target tree name is retained after the merge. Change the preferred tree name to the new tree name.

HINT: To minimize the number of NET.CFG files you need to update, designate the tree with the most client workstations as the target tree because the final tree retains the name of the target tree. Or rename the tree after the merge operation so that the final tree name corresponds to the majority of the client workstations' NET.CFG files. For more information, see .

The Access Control List (ACL) for the Tree object of the source tree is preserved. Therefore, the rights of the source tree's user Admin to the Tree object are still valid.

After the merge is complete, both admin users still exist and are uniquely identified by different container objects.

For security reasons, you might want to delete one of the two Admin User objects or restrict the rights of the two objects.

Grafting a Single Server Tree

Understanding Grafting

The Graft Tree option lets you graft a single-server source tree's Tree object under a container specified in the target tree. After the graft is completed, the source tree receives the target tree's name.

During the graft, DSMERGE changes the object class of the source tree's Tree object to Domain and makes a new partition. The new Domain object is the partition root for the new partition. All the objects under the source tree's Tree object are located under the Domain object.

The target tree's administrator has rights to the resulting tree's root container and therefore has rights to the source tree's grafted root.

NOTE: It may take up to several hours for the inherited rights to be recalculated and become effective. This time will vary based on the tree's complexity, size, and number of partitions.

The source tree's administrator has rights only in the newly created domain object.

Figure 34 on page 275 and Figure 35 on page 276 illustrate the effect when you graft a tree into a specific container.

Figure 34   eDirectory Trees before a Graft
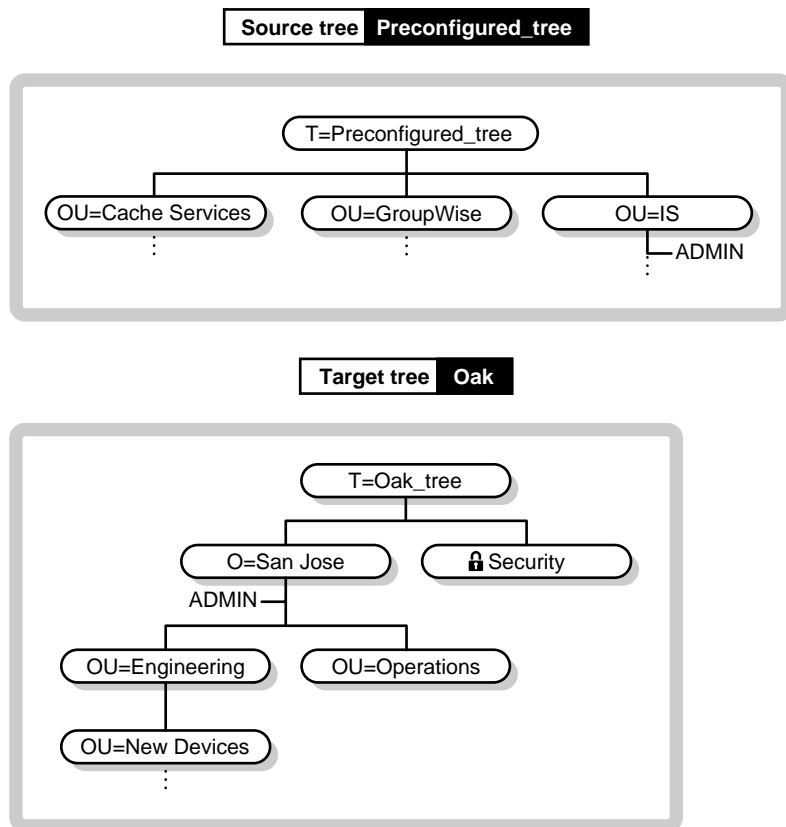
Figure 35   eDirectory Tree after a Graft

Source tree grafted under the New Devices container

Context Name Changes

After the graft of the source tree into the target tree container, the distinguished names for objects in the source tree will be appended with the source tree's name followed by the distinguished name of the target tree's container name where the source tree was merged. The relative distinguished name will remain the same.

For example, if you are using dot delimiters, the typeful name for Admin in the Preconfigured_tree (source tree) is:

```
CN=Admin.OU=IS.T=Preconfigured_tree
```

After the Preconfigured_tree is merged into the New Devices container in the Oak_tree, the typeful name for Admin is:

```
CN=Admin.OU=IS.DC=Preconfigured_tree.OU=Newdevices
  .OU=Engineering.O=Sanjose.T=Oak_tree.
```

NOTE: The distinguished name maximum character length is 256 characters. This limitation is particularly important when you are grafting the root of one tree into a container near the bottom of the target tree.

The last dot following Oak_tree (Oak_tree.) indicates that the last element in the distinguished name is the tree name. If you leave off the trailing dot, then leave off the tree name.

Preparing the Source and Target Trees

Before initiating the graft operation, ensure that the state of all of the servers affected by the operation is stable. Table 59 provides recommendations for preparing the source and target trees before grafting.

Table 59   **Recommendations for Preparing Source and Target Trees before Grafting**

Prerequisite	Required Action
WANMAN should be turned off on all servers that hold a replica of the source tree's Tree partition or the target tree's Tree partition.	Review your WANMAN policy so that WAN communication restrictions do not interfere with the merge operation. If required, turn WANMAN off before initiating the merge operation.
The source tree must have only one server.	Remove all but one server from the source tree.
No aliases or leaf objects can exist at the source tree's Tree object.	Delete any aliases or leaf objects at the source tree's Tree object.

Prerequisite	Required Action
No similar names can exist in the graft container.	Rename objects under the target tree graft container or rename the source tree.
	Move objects from one of the containers to a different container in its tree if you don't want to rename objects, then delete the empty container before running DSMERGE. For more information, see Chapter 4, "Managing Objects," on page 143.
	You can have identical container objects in both trees if they are not immediately subordinate to the same parent object. Objects are uniquely identified by their immediate container object.
The eDirectory version for both the source tree and target tree container must be 8.51 SP2a or later.	DSMERGE will search for the appropriate version of eDirectory. If an acceptable version isn't found, DSMERGE will return an error. You can get the latest version of eDirectory from the product download page (http://www.novell.com/download/).
The container where you will join the target tree is in a partition that has no replicas (a single-server partition).	If the target container has multiple replicas:
	◆ Make the partition associated with this container the master replica, then delete other replicas, or
	◆ Split the target tree graft container in to a separate partition and then remove replicas.
	After the graft is complete, the partition association can be re-established.

Prerequisite	Required Action
The schema on both the source and target trees must be the same.	Run the Graft option in DSMERGE. If reports indicate schema problems, run DSREPAIR on the target tree to import the schema from the source tree. The graft operation automatically imports the schema from the target tree to the source tree. Run DSMERGE again.
Only one tree can have a security container subordinate to the tree root.	If both the source and target tree have the security container, remove one container as explained in Appendix A, "NMAS Considerations," on page 527.
The source tree's time reference must be reconfigured.	Generally the source tree should be set as a secondary server configured to get its time source from a server in the target tree. To reconfigure Timesync, see Configuring Timesync on Servers in the Network Time Management Administration Guide.

Containment Requirements for Grafting

To graft a source tree into a target tree container requires that the target tree container be prepared to accept the source tree. The target tree container must be able to contain an object of the class domain. If there is a problem with containment, error -611 Illegal Containment will occur during the graft operation.

Use the information in Table 60 to determine if you need to run DSREPAIR to modify containment lists.

Table 60   Target and Source Tree Container Requirements

Target Tree Container Requirements	The target tree container object must include the domain object in its containment list.
	You can check this using iMonitor > Schema. If the containment list does not include Domain, run DSREPAIR to make schema enhancements.
Source Tree Requirements	The graft operation changes the source tree root from the class Tree Root to the class Domain. All of the object classes that are subordinate to the Tree must be able to be contained by the class Domain according to the schema rules.
	You can check this using iMonitor > Schema. If the containment list does not include Domain, run DSREPAIR to make schema enhancements.

If containment requirements aren't met, run DSREPAIR to correct the schema.

1 Start NDSCONSOLE by running NDSCONS.EXE.

2 Select DSREPAIR.DML > click Start.

3 Click Schema > Optional Schema Enhancements.

The schema is updated and results are displayed on screen.

Grafting the Source and Target Tree

After you ensure that prerequisites are met, use DSMERGE to perform the graft as explained in the following procedure:

1 Start NDSCONSOLE by running NDSCONS.EXE.

2 Select DSMERGE.DML > click Start.

3 Choose Operations > Graft a Single Server Tree.

4 Enter the required data including user name and password for both the source and target tree and the distinguished name of the target tree container, for example,
   New Devices.Engineering.San Jose.Oak_tree

5 Choose Ok to start the graft.

Using ndsmerge for Linux or Solaris

You can use the ndsmerge utility on Linux or Solaris systems to combine two eDirectory trees so that the combined tree can be accessed by the clients of both the trees. The ndsmerge options lets you:

 Merge two trees

 Check the time synchronization of all the servers hosting the root replica of the tree

 Change the name of the source tree to the name of the target tree

 List all the servers hosting the root replica present in the tree, along with the time synchronization status

The following sections provide information that help you perform ndsmerge operations on Linux or Solaris:

Prerequisites for Performing ndsmerge Operations

Meet the following prerequisites before merging two eDirectory trees:

❑ Identify the source and destination trees. While merging, servers on the source tree will combine with the target tree and the source tree will cease to exist. If a target-container is not specified, the trees will be merged at the Tree object level by default. This may affect users logging in, if the preferred tree is specified in the configuration file.

❑ Ensure that both the trees are operating properly and that the replicas are synchronizing without any errors.

❑ Ensure that all servers in the tree are synchronizing from the same time source.

❑ Trees cannot be merged if the schema on the source and target tree do not match. If you have installed an application that modified the schema on a tree, install the same application on the other tree so that the schema on both trees match. Alternatively, you can use the Import Remote Schema option of the ndsrepair utility to adjust the schema between two trees. To do so, run the following command on both the trees:

ndsrepair - S -Ad

❑ Run **ndsrepair -U** on both the trees before starting the merge operation. This operation verifies that all replicas in a tree are synchronizing properly, identifies synchronization errors, and corrects them.

Merging eDirectory Trees on Linux or Solaris Systems

To merge two eDirectory trees:

1 Use the following syntax:

```
ndsmerge [-m target-tree target-admin source-admin
[target_container]] [-c] [-t] [-r target-tree source-
admin]
```

Table 61   ndsmerge Options

Option	Description

-m	Merges two trees. You need to specify the following attributes with this option:
	
	• target-tree
	
	  Name of the target tree.
	
	• target-admin
	
	  Name with full context of the user with administration rights to the target tree.
	
	• source-admin
	
	  Name with full context of the user with administration rights to the source tree.
	
	• target-container
	
	  Name with full context of the container object on the target tree to which the Tree object of the source tree has to be combined. If you choose to specify a value to this parameter, ensure that the source tree has a single server.

-c	Lists all the servers present in the tree with the synchronization status. This option can be used before merging trees to detect any problems before starting the merge operation.
-t	Checks the time synchronization of all the servers in the tree. Use this option on the server having the master of the Tree partition. This will list the servers in the tree, along with the time synchronization status.

Option	Description
-r	Changes the name of the source tree to the name of the target tree. Use this option when you are merging two trees with the same name. You have to log in to the tree to change the name of the tree. For this operation to succeed, ensure that you run ndsmerge on the server having the master of the tree's Tree partition. All the servers in the tree will merge with the new tree name. You need to specify the following attributes with this option:
	◆ target-tree
	The new name for the tree.
	◆ source-admin
	The name with full context of the user with administration rights to the tree.

After the merge operation is completed, ndsmerge waits for the new replica of the root partition to synchronize with the root server of the source tree. If any synchronization problems are reported, you need to run the ndsrepair utility to rectify synchronization problems. For more information, see "Using ndsrepair" on page 503.

9 WAN Traffic Manager

WAN Traffic Manager (WTM) lets you manage replication traffic across WAN links, reducing network costs. WAN Traffic Manager is installed during the Novell® eDirectory™ installation and consists of three elements.

- ◆ WTM

    WTM resides on each server in the replica ring. Before eDirectory sends server-to-server traffic, WTM reads a WAN traffic policy and determines whether that traffic will be sent.

- ◆ WAN traffic policies

    These are rules that control the generation of eDirectory traffic. WAN traffic policies are text stored as an eDirectory property value on a Server object, a LAN Area object, or both.

- ◆ WANMAN ConsoleOne™ snap-in

    This snap-in is the interface to WTM. It lets you create or modify policies, create LAN Area objects, and apply policies to LAN areas or servers. When WTM is installed (as part of the eDirectory installation), the schema includes a LAN Area object and a WAN Traffic Manager page on the Server object.

WAN Traffic Manager (WTM.NLM on NetWare® or WTM.DLM on Windows* NT*) must reside on each server whose traffic you want to control. If a partition's replica ring includes servers on both sides of a wide area link, you should install WAN Traffic Manager on all servers in that replica ring.

IMPORTANT: WAN Traffic Manager is not supported on Linux* or Solaris* platforms.

Understanding WAN Traffic Manager

Network directories, such as eDirectory, create server-to-server traffic. If this traffic crosses wide area network (WAN) links unmanaged, it can needlessly increase costs and overload slow WAN links during high-usage periods.

WAN Traffic Manager lets you control server-to-server traffic (over WAN links) generated by eDirectory and control eDirectory traffic between any servers in an eDirectory tree. WTM can restrict traffic based on cost of traffic, time of day, type of eDirectory operations, or any combination of these.

For example, you might restrict eDirectory traffic over a WAN link during high-usage times. This shifts high-bandwidth activities to off-hours. You might also limit replica synchronization traffic to times when rates are low to reduce costs.

WAN Traffic Manager controls only periodic events initiated by eDirectory, such as replica synchronization. It does not control events initiated by administrators or users, nor does it control non-eDirectory server-to-server traffic such as time synchronization.

The eDirectory processes listed in Table 62 generate server-to-server traffic:

Table 62   **Novell eDirectory Processes that Generate Server-to-Server Traffic**

Process	Description
Replica synchronization	Ensures that changes to eDirectory objects are synchronized among all replicas of the partition. This means that any server that holds a copy of a given partition must communicate with the other servers to synchronize a change.
	Two types of replica synchronization can occur:
	◆ Immediate sync occurs after any change to an eDirectory object or any addition or deletion of an object in the directory tree.
	◆ Slow sync occurs for specific changes to an eDirectory object that are repetitive and common to multiple objects, such as changes to login properties. Some examples of this are updates to Login Time, Last Login Time, Network Address, and Revision properties when a user logs in or out.
	The slow sync process runs only in the absence of an immediate sync process. By default, immediate sync runs 10 seconds after any change is saved and slow sync runs 22 minutes after other changes are made.
Schema synchronization	Ensures that the schema is consistent across the partitions in the directory tree and that all schema changes are updated across the network.
	This process runs once every 4 hours by default.
Heartbeat	Ensures that directory objects are consistent among all replicas of a partition. This means that any server with a copy of a partition must communicate with the other servers holding the partition to check the consistency.
	This process runs by default once every 30 minutes on every server that contains a replica of a partition.

Process	Description
Limber	Ensures that a server's replica pointer table is updated when that server's name or address is changed. Such changes occur when: • The server is rebooted with a new server name or IPX™ internal address in the AUTOEXEC.NCF file. • An address is added for an additional protocol. When a server is booted, the limber process compares the server's name and IPX address with those stored in the replica pointer table. If either is different, eDirectory automatically updates all replica pointer tables that contain a listing of that server. The limber process also checks that the tree name is correct for each server in a replica ring. Limber runs 5 minutes after the server boots up and then every 3 hours.
Backlink	Verifies external references, which are pointers to eDirectory objects that are not stored in the replicas on a server. The backlink process normally runs 2 hours after the local database is opened and then every 13 hours.
Connection management	Servers in a replica ring require a highly-secure connection for transferring NCP packets. These secure connections, called virtual client connections, are established by the connection management process. The connection management process might also need to establish a virtual client connection for schema synchronization or backlink processes. Time synchronization might also require such a connection, depending on the configuration of time services.
Server status check	Each server without a replica initiates a server status check. It establishes a connection to the nearest server that holds a writable replica of the partition containing the Server object. The server status check runs every 6 minutes.

LAN Area Objects

A LAN Area object lets you easily administer WAN traffic policies for a group of servers. Once you create a LAN Area object, you can add servers to or remove servers from the LAN Area object. When you apply a policy to the LAN Area, that policy applies to all the servers in the LAN Area.

You should create a LAN Area object if you have multiple servers in a LAN that is connected to other LANs by wide area links. If you do not create a LAN Area object, you must manage each server's WAN traffic individually.

Creating a LAN Area Object

1 In ConsoleOne, right-click the container you want to create LAN Area object in.

2 Click New > Object.

3 Under Class, click WANMAN:LAN Area > OK.

4 Enter a name for the object > click OK.

5 Continue with one of the topics below:

"Adding Servers to a LAN Area Object" on page 289

"Applying WAN Policies" on page 292

Adding Servers to a LAN Area Object

A server can belong to only one LAN Area object. If the server you are adding already belongs to a LAN Area object, the server will be removed from that object and added to the new object.

1 In ConsoleOne, right-click a LAN Area object.

2 Click Properties > Members.

3 Click Add.

4 Select the server you want > click OK.

5 Repeat Step 3 through Step 4 for each server you want to add.

To apply a WAN policy to the LAN Area object, thereby applying the policy to all the servers in the group, see "Applying WAN Policies" on page 292.

6 Click Apply > OK.

Adding Additional Information to a LAN Area Object

You can add descriptive information to a LAN Area object.

1 In ConsoleOne, right-click a LAN Area object.

2 Click Properties > General.

3 Add the Owner, Description, Location, Department, and Organization information you want.

4 Click Apply > OK.

WAN Traffic Policies

A WAN traffic policy is a set of rules that control the generation of eDirectory traffic. These rules are created as text and are stored as an eDirectory property value on the Server object, the LAN Area object, or both. The policy is interpreted according to a simple processing language.

You can apply policies to individual servers or you can create LAN Area objects and assign several servers to one of these objects. Any policy that is applied to the LAN Area object is automatically applied to all servers that are assigned to the object.

WAN Traffic Manager comes with several predefined policy groups. You can use these policies as they are, modify them to meet your needs, or write new policies.

Predefined Policy Groups

Table 63 lists groups of predefined policies with similar functions:

Table 63   **Predefined Policy Groups with Similar Functions**

Policy Group	Description
1-3AM.WMG	Limits the time traffic is sent to between 1 a.m. and 3 a.m.
7AM-6PM.WMG	Limits the time traffic is sent to between 7 a.m. and 6 p.m.
COSTLT20.WMG	Only allows traffic to be sent that has a cost factor below 20.
IPX.WMG	Allows only IPX traffic.
NDSTTYPS.WMG	Provides sample policies for various eDirectory traffic types.
ONOSPOOF.WMG	Allows only existing WAN connections to be used.
OPNSPOOF.WMG	Allows only existing WAN connections to be used but assumes that a connection that hasn't been used for 15 minutes is being spoofed and should not be used.
SAMEAREA.WMG	Allows traffic only in the same network area.
TCPIP.WMG	Allows only TCP/IP traffic.
TIMECOST.WMG	Restricts all traffic to between 1 a.m. and 1:30 a.m. but allows servers in the same location to talk continuously.

For detailed information on the predefined policy groups and their individual policies, see "WAN Traffic Manager Policy Groups" on page 299.

Applying WAN Policies

You can apply WAN policies to an individual server or to a LAN Area object. Policies applied to an individual server manage eDirectory traffic for that server only. Policies applied to a LAN Area object manage traffic for all servers that belong to the object.

WAN Traffic Manager will look in WANMAN.INI for a WAN policy groups section, which contains a *key = values* statement. *Key* is the policy name displayed in the snap-in and *value* is the path to the text files containing delimited policies.

Applying WAN Polices to a Server

1 In ConsoleOne, right-click the Server object that you want to apply a policy to.

2 Click Properties > WAN Traffic Manager-Policies.

3 Click Load > select the policy group you want.

See for more information.

4 Click Open.

The Policies list box displays a list of the policies loaded from the policy group.

5 To review the policies, select the policy > click Edit.

You can read what the policy does, make changes to the policy, or click Check to check for errors in the policy.

6 Click Save if you made any changes.

or

Click Cancel to return to the WAN Traffic Manager-Policies page.

7 To remove any policies that you don't want, select a policy > click Delete > Yes.

8 Click Apply > OK.

Applying WAN Policies to a LAN Area Object

1 In ConsoleOne, right-click the LAN Area object that you want to apply a policy to.

2 Click Properties > Policies.

3 Click Load, then select the policy group you want.

See "Predefined Policy Groups" on page 291 for more information.

4 Click Open.

The Policies list box displays a list of the policies loaded from the policy group.

5 To review the policies, select the policy > click Edit.

You can read what the policy does, make changes to the policy, or click Check to check for errors in the policy.

6 Click Save if you made any changes.

or

Click Cancel to return to the WAN Traffic Manager-Policies page.

7 To remove any policies that you don't want, select a policy > click Delete > Yes.

8 Click Apply > OK.

Modifying WAN Policies

You can modify one of the predefined policy groups included with WAN Traffic Manager to meet your own needs. You can also modify a policy you wrote yourself.

Modifying WAN Policies Applied to a Server

1 In ConsoleOne, right-click the Server object that contains the policy you want to edit.

2 Click Properties > WAN Traffic Manager-Policies.

3 Select the policy you want > click Edit.

4 Edit the policy to meet your needs.

To understand the structure of a WAN policy, see "WAN Policy Structure" on page 320.

To understand the syntax of a WAN policy, see "Construction Used within Policy Sections" on page 325.

5 Click Check to identify errors in syntax or structure.

WAN Traffic Manager will not run policies with errors.

6 Click Save if you made any changes.

or

Click Cancel to return to the WAN Traffic Manager-Policies page.

7 To remove any policies that you don't want, select a policy > click Delete > Yes.

8 Click Apply > OK.

Modifying WAN Policies Applied to a LAN Area Object

1 In ConsoleOne, right-click the LAN Area object that contains the policy you want to edit.

2 Click Properties > Policies.

3 Select the policy you want > click Edit.

4 Edit the policy to meet your needs.

To understand the structure of a WAN policy, see "WAN Policy Structure" on page 320.

To understand the syntax of a WAN policy, see "Construction Used within Policy Sections" on page 325.

5 Click Check to identify errors in syntax or structure.

WAN Traffic Manager will not run policies with errors.

6 Click Save if you made any changes or Cancel to return to the WAN Traffic Manager-Policies page.

7 To remove any policies that you don't want, select a policy > click Delete > Yes.

8 Click Apply > OK.

Renaming an Existing Policy

1 In ConsoleOne, right-click a Server or LAN Area object.

2 Click Properties > WAN Traffic Manager-Policies (for a Server object) or Policies (for a LAN Area object).

3 Select the policy you want > click Rename.

4 Enter a new name for the policy.

The name must be a fully distinguished name.

Creating New WAN Policies

You can write a WAN policy for a Server object or a LAN Area object. Policies written for an individual server manage eDirectory traffic for that server only, while policies written for a LAN Area object manage traffic for all servers that belong to the object.

Creating a WAN Policy for a Server Object

1 In ConsoleOne, right-click the Server object that you want to add a new policy to.

2 Click Properties > WAN Traffic Manager-Policies.

3 Click Add > enter a name for the new policy.

   The name you enter for the new policy should be a fully distinguished name.

4 Enter the necessary information in the Policy list box.

   To understand the structure of a WAN policy, see "WAN Policy Structure" on page 320.

   To understand the syntax of a WAN policy, see "Construction Used within Policy Sections" on page 325.

   You might also look at one or more predefined policies as examples. In many cases it is easier to modify an existing policy than to write an entirely new one.

5 Click Check to identify errors in syntax or structure.

   WAN Traffic Manager will not run policies with errors.

6 Click Save to return to the WAN Traffic Manager-Policies page.

7 Click Apply > OK.

Creating a WAN Policy for a LAN Area Object

1 In ConsoleOne, right-click the LAN Area object that you want to add a new policy to.

2 Click Properties > Policies.

3 Click Add > enter a name for the new policy.

   The name you enter for the new policy should be a fully distinguished name.

4 Enter the necessary information in the Policy list box.

To understand the structure of a WAN policy, see "WAN Policy Structure" on page 320.

To understand the syntax of a WAN policy, see "Construction Used within Policy Sections" on page 325.

You might also look at one or more predefined policies as examples. In many cases it is easier to modify an existing policy than to write an entirely new one.

5 Click Check to identify errors in syntax or structure.

WAN Traffic Manager will not run policies with errors.

6 Click Save return to the WAN Traffic Manager-Policies page.

7 Click Apply > OK.

Limiting WAN Traffic

WAN Traffic Manager comes with two predefined WAN Policy groups that limit traffic to specific hours. (For more information, see "1-3AM.WMG" on page 299 and "7AM-6PM.WMG" on page 300.) You can modify these policies to limit traffic to any span of hours you select.

The instructions below are for modifying the 1:00 a.m. to 3:00 a.m. group, but you can use the same steps to accomplish the same thing with the 7:00 a.m.to 6:00 p.m. group.

1 In ConsoleOne, right-click a Server or LAN Area object.

2 Click Properties > WAN Traffic Manager-Policies (for a Server object) or Policies (for a LAN Area object).

3 Click Load > select 1-3AM.WMG > click Open.

The Policies list box displays the policies in the group. Two policies will load: 1-3 am and 1-3 am, NA. If you plan to manage backlink traffic, you will need to follow the steps below for both 1-3 am and 1-3 am, NA.

4 In the Policies list box, select the 1-3 am policy > click Edit.

The policy is displayed in a simple text editor, which allows you to make changes. For example, if you want to limit traffic to 2:00 a.m. to 5:00 p.m. rather than from 1:00 a.m. to 3:00 a.m., make the following changes:

```
/* This policy limits all traffic to between 2 and 5 pm */
```

```
LOCAL BOOLEAN Selected;
SELECTOR
  Selected := Now.hour >= 2 AND Now.hour < 17;
  IF Selected THEN
    RETURN 50; /* between 2am and 5pm this policy has a
 high priority */
  ELSE
    RETURN 1;  /* return 1 instead of 0 in case there are
 no other policies */
              /* if no policies return > 0, WanMan assumes
 SEND */
  END
END
PROVIDER
  IF Selected THEN
    RETURN SEND; /* between 2am and 5pm, SEND */
  ELSE
    RETURN DONT_SEND; /* other times, don't */
  END
END
```

In the comment lines (set off with /* and */), the hour can be designated using a.m. and p.m. In the active code, however, it must be designated using 24-hour format. In that case, 5:00 p.m. becomes 17.

To better understand the structure of a WAN policy, see "WAN Policy Structure" on page 320.

To better understand the syntax of a WAN policy, see "Construction Used within Policy Sections" on page 325.

5 Click Check to identify errors in syntax or structure.

WAN Traffic Manager will not run policies with errors.

6 Click Save.

7 If you want to keep the original 1-3 a.m.policy, add the new policy under a different name.

8 Click Apply > OK.

Assigning Cost Factors

Cost factors let WAN Traffic Manager compare the cost of traffic with certain destinations, then manage the traffic using WAN policies. WAN policies use cost factors to determine the relative expense of WAN traffic. You can then use this information in determining whether to send traffic.

A cost factor is expressed as expense per unit of time. It can be in any units as long as the same units are used consistently in each WAN traffic policy. You can use dollars per hour, cents per minute, yen per second, or any other ratio of expense to time, as long as you use that ratio exclusively.

You can assign destination cost factors representing the relative expense of traffic to particular address ranges. Therefore, you can assign cost for an entire group of servers in one declaration. You can also assign a default cost factor to be used when no cost is specified for a destination.

If no cost is assigned for the destination, the default cost is used. If you have specified no default cost for the server or LAN Area object, a value of -1 is assigned.

For information about a sample policy that restricts traffic based on cost factor, see "COSTLT20.WMG" on page 300.

For information about how to modify a policy, see "Modifying WAN Policies" on page 293.

Assigning Default Cost Factors

1 In ConsoleOne, right-click a Server or LAN Area object.

2 Click Properties > WAN Traffic Manager-Cost (for a Server object) or Cost (for a LAN Area object).

3 Enter a cost in the Default Cost field.

The cost must be a non-negative integer. If supplied, the default cost will be assigned to all destinations in the Server or LAN Area object that do not fall within a destination address range with an assigned cost. For example, you might specify the cost in monetary units, such as dollars, or in packets per second.

4 Click Apply > OK.

Assigning a Cost to a Destination Address Range

1 In ConsoleOne, right-click a Server or LAN Area object.

2 Click Properties > WAN Traffic Manager-Cost (for a Server object) or Cost (for a LAN Area object).

3 Click Add TCP/IP or Add IPX.

4 Specify the start address and stop address of the range, in the appropriate format for TCP/IP or IPX.

5 Specify the cost as a non-negative integer.

6 Click OK.

7 Click Apply > OK.

Before new cost factors become effective, you must either enter the WANMAN REFRESH IMMEDIATE command at the server console or reload WTM.

WAN Traffic Manager Policy Groups

WAN Traffic Manager comes with the following predefined policy groups. For more information on applying policy groups in ConsoleOne, see "Applying WAN Policies" on page 292.

1-3AM.WMG

The policies in this group limit the time traffic can be sent to between 1 a.m. and 3 a.m. There are two policies.

◆ 1 - 3 am, NA

This policy limits the checking of backlinks, external references, and login restrictions, the running of janitor or limber, and schema synchronization to these hours.

◆ 1 - 3 am

This policy limits all other traffic to these hours.

To restrict all traffic to these hours, both policies must be applied.

7AM-6PM.WMG

The policies in this group limit the time traffic can be sent to between 7 a.m. and 6 p.m. There are two policies.

- ◆ 7 am - 6 pm, NA

  This policy limits the checking of backlinks, external references, and login restrictions, the running of janitor or limber, and schema synchronization to these hours.

- ◆ 7 am - 6 pm

  This policy limits all other traffic to these hours.

To restrict all traffic to these hours, both policies must be applied.

COSTLT20.WMG

The policies in this group only allow traffic to be sent that has a cost factor below 20. There are two policies.

- ◆ Cost < 20, NA

  This policy prevents the checking of backlinks, external references, and login restrictions, the running of janitor or limber, and schema synchronization unless the cost factor is less than 20.

- ◆ Cost < 20

  This policy prevents all other traffic unless the cost factor is less than 20.

To prevent all traffic with a cost factor of 20 or more, both policies must be applied.

IPX.WMG

The policies in this group allow only IPX traffic. There are two policies.

- ◆ IPX, NA

  This policy prevents the checking of backlinks, external references, and login restrictions, the running of janitor or limber, and schema synchronization unless the traffic that is generated is IPX.

- ◆ IPX

  This policy prevents all other traffic unless the traffic is IPX.

To prevent all non-IPX traffic, both policies must be applied.

NDSTTYPS.WMG

The policies in this group are sample policies for various eDirectory traffic types. They contain the variables eDirectory passes in a request of this type.

Sample Catch All with Addresses

This is a sample policy for traffic types with addresses.

Sample Catch All without Addresses

This is a sample policy for traffic types without addresses.

Sample NDS_BACKLINKS

Before eDirectory checks any backlinks or external references, it queries WAN Traffic Manager to see if this is an acceptable time for this activity. NDS_BACKLINKS does not have a destination address; it requires a NO_ADDRESSES policy. If WAN Traffic Manager returns DONT_SEND, backlink checking will be put off and rescheduled. The following variables are supplied.

◆ *Last* (Input Only, Type TIME)

The time of the last round of backlink checking since eDirectory started. When eDirectory starts, *Last* is initialized to 0. If NDS_BACKLINKS returns SEND, *Last* is set to the current time after eDirectory finishes backlinking.

◆ *Version* (Input Only, Type INTEGER)

The version of eDirectory.

◆ *ExpirationInterval* (Output Only, Type INTEGER)

The expiration interval for all connections created while backlinking.

Table 64

Value	Description
<0, 0	Use the default expiration interval (default).
>0	Expiration interval to be assigned to this connection.

◆ *Next* (Output Only, Type TIME)

This variable indicates when eDirectory should schedule the next round of backlink checking.

Table 65

Value	Description
In past, 0	Use the default scheduling.
In future	Time when backlinking should be scheduled.

◆ *CheckEachNewOpenConnection* (Output Only, Type INTEGER)

This variable tells eDirectory what to do if it needs to create a new connection while doing backlinking.

CheckEachNewOpenConnection is initialized to 0.

Table 66

Value	Description
0	Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default).
1	Call WAN Traffic Manager and let the policies decide whether to allow the connection.
2	Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail.

◆ *CheckEachAlreadyOpenConnection* (Output Only, Type INTEGER)

This variable tells eDirectory what to do if it needs to reuse a connection it believes is already open while doing backlinking. *CheckEachAlreadyOpenConnection* is initialized to 0.

Table 67

Value	Description
0	Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default).
1	Call WAN Traffic Manager and let the policies decide whether to allow the connection.
2	Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail.

Sample NDS_BACKLINK_OPEN

NDS_BACKLINK_OPEN is a traffic type that is used only if either *CheckEachNewOpenConnection* or *CheckEachAlreadyOpenConnection* was set to 1 during the corresponding NDS_BACKLINKS query.

This query is generated whenever *CheckEachNewOpenConnection* is 1 and eDirectory needs to open a new connection for backlinking or *CheckEachAlreadyOpenConnection* is 1 and eDirectory needs to reuse an already existing connection.

- *Version* (Input Only, Type INTEGER)

  The version of eDirectory.

- *ExpirationInterval* (Input and Output, Type INTEGER)

  If ConnectionIsAlreadyOpen is TRUE, ExpirationInterval will be set to the expiration interval already set on the existing connection. Otherwise it will be set to the ExpirationInterval assigned in the NDS_BACKLINKS query. A 0 value indicates that the default (2 hours) should be used. On exit, the value of this variable is assigned as the expiration interval for the connection.

Table 68

Value	Description
<0, 0	Use the default expiration interval (default).
>0	Expiration interval to be assigned to this connection.

- *ConnectionIsAlreadyOpen* (Input Only, Type BOOLEAN)

  This variable is TRUE if eDirectory can reuse an existing connection and FALSE if it needs to create a new connection.

Table 69

Value	Description
TRUE	eDirectory determines that it already has a connection to this address and can reuse that connection.
FALSE	eDirectory does not have a connection to this address and must create one.

◆ *ConnectionLastUsed* (Input Only, Type TIME)

If *ConnectionIsAlreadyOpen* is TRUE, then *ConnectionLastUsed* is the last time that a packet was sent from eDirectory using this connection. Otherwise, it will be 0.

Table 70

Value	Description
TRUE	*ConnectionLastUsed* is the time that eDirectory last sent a packet on this connection.
FALSE	*ConnectionLastUsed* will be 0.

Sample NDS_CHECK_LOGIN_RESTRICTION

Before eDirectory checks a login restriction, it queries WAN Traffic Manager to see if this is an acceptable time for this activity. The traffic type NDS_CHECK_LOGIN_RESTRICTIONS does not have a destination address; it requires a NO_ADDRESSES policy. If WAN Traffic Manager returns DONT_SEND, the check will error out. The following variables are supplied:

◆ *Version* (Input Only, Type INTEGER)

The version of eDirectory.

◆ *Result* (Output Only, Type INTEGER)

If the result of NDS_CHECK_LOGIN_RESTRICTIONS is DONT_SEND, then the values in Table 71 will be returned to the operating system.

Table 71

Value	Description
0	Login is allowed.
1	Login is not allowed during the current time block.
2	Account is disabled or expired.
3	Account has been deleted.

◆ *ExpirationInterval* (Output Only, Type INTEGER)

The expiration interval that should be assigned to this connection.

Table 72

Value	Description
<0, 0	Use the default expiration interval (default).
>0	Expiration interval to be assigned to this connection.

◆ *CheckEachNewOpenConnection* (Output Only, Type INTEGER)

Table 73

Value	Description
0	Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default).
1	Call WAN Traffic Manager and let the policies decide whether to allow the connection.
2	Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail.

◆ *CheckEachAlreadyOpenConnection* (Output Only, Type INTEGER)

Table 74

Value	Description
0	Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default).
1	Call WAN Traffic Manager and let the policies decide whether to allow the connection.
2	Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail.

Sample NDS_CHECK_LOGIN_RESTRICTION_OPEN

NDS_CHECK_LOGIN_RESTRICTION_OPEN is used only if either *CheckEachNewOpenConnection* or *CheckEachAlreadyOpenConnection* was set to 1 during the corresponding NDS_CHECK_LOGIN_RESTRICTIONS query. This query is generated whenever *CheckEachNewOpenConnection* is 1 and eDirectory needs to:

- ◆ Open a new connection before running limber
- ◆ Open a new connection before checking the login restriction
- ◆ Reuse an already existing connection

The following variables are provided:

- ◆ *Version* (Input Only, Type INTEGER)

  The version of eDirectory.

- ◆ *ExpirationInterval* (Input and Output, Type INTEGER)

Table 75

Value	Description
<0, 0	Use the default expiration interval (default).
>0	Expiration interval to be assigned to this connection.

- ◆ *ConnectionIsAlreadyOpen* (Input Only, Type BOOLEAN)

Table 76

Value	Description
TRUE	eDirectory determines that it already has a connection to this address and can reuse that connection.
FALSE	eDirectory does not have a connection to this address and must create one.

◆ *ConnectionLastUsed* (Input Only, Type TIME)

If *ConnectionIsAlreadyOpen* is TRUE, then *ConnectionLastUsed* is the last time that a packet was sent from eDirectory using this connection. Otherwise, it will be 0.

Table 77

Value	Description
TRUE	*ConnectionLastUsed* is the time that eDirectory last sent a packet on this connection.
FALSE	*ConnectionLastUsed* will be 0.

Sample NDS_JANITOR

Before eDirectory runs the janitor, it queries WAN Traffic Manager to see if this is an acceptable time for this activity. The NDS_JANITOR does not have a destination address; it requires a NO_ADDRESSES policy. If WAN Traffic Manager returns DONT_SEND, janitor work will be put off and rescheduled. The following variables are supplied:

◆ *Last* (Input Only, Type TIME)

The time of the last round of janitor work since eDirectory started. When eDirectory starts, *Last* is initialized to 0. If NDS_JANITOR returns SEND, *Last* is set to the current time after eDirectory finishes the janitor.

◆ *Version* (Input Only, Type INTEGER)

The version of eDirectory.

◆ *ExpirationInterval* (Output Only, Type INTEGER)

The expiration interval for all connections created while running the janitor.

Table 78

Value	Description
<0, 0	Use the default expiration interval (default).
>0	Expiration interval to be assigned to this connection.

◆ *Next* (Output Only, Type TIME)

This variable indicates when eDirectory should schedule the next round of janitor work.

Table 79

Value	Description
In the past, 0	Use the default scheduling.
In the future	Time when the janitor should be scheduled.

◆ *CheckEachNewOpenConnection* (Output Only, Type INTEGER)

This variable tells eDirectory what to do if it needs to create a new connection while running the janitor.

CheckEachNewOpenConnection is initialized to 0.

Table 80

Value	Description
0	Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default).
1	Call WAN Traffic Manager and let the policies decide whether to allow the connection.
2	Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail.

◆ *CheckEachAlreadyOpenConnection* (Output Only, Type INTEGER)

This variable tells eDirectory what to do if it needs to reuse a connection it determines is already open while running the janitor. *CheckEachAlreadyOpenConnection* is initialized to 0.

Table 81

Value	Description
0	Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default).
1	Call WAN Traffic Manager and let the policies decide whether to allow the connection.
2	Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail.

Sample NDS_JANITOR_OPEN

NDS_JANITOR_OPEN is used only if either *CheckEachNewOpenConnection* or *CheckEachAlreadyOpenConnection* was set to 1 during the corresponding NDS_JANITOR query. This query is generated whenever *CheckEachNewOpenConnection* is 1 and eDirectory needs to open a new connection before doing backlinking, or *CheckEachAlreadyOpenConnection* is 1 and eDirectory needs to reuse an already existing connection.

◆ *Version* (Input Only, Type INTEGER)

The version of eDirectory.

◆ *ExpirationInterval* (Input and Output, INTEGER)

If *ConnectionIsAlreadyOpen* is TRUE, ExpirationInterval will be set to the expiration interval already set on the existing connection. Otherwise, it will be set to the ExpirationInterval assigned in the NDS_JANITOR query. A 0 value indicates that the default (2 hours, 10 seconds) should be used. On exit, the value of this variable is assigned as the expiration interval for the connection.

Table 82

Value	Description
<0, 0	Use the default expiration interval (default).
>0	Expiration interval to be assigned to this connection.

◆ *ConnectionIsAlreadyOpen* (Input Only, Type BOOLEAN)

This variable is TRUE if eDirectory needs to reuse an existing connection and FALSE if it needs to create a new connection.

Table 83

Value	Description
TRUE	eDirectory determines that it already has a connection to this address and can reuse that connection.
FALSE	eDirectory does not have a connection to this address and must create one.

◆ *ConnectionLastUsed* (Input Only, Type TIME)

If *ConnectionIsAlreadyOpen* is TRUE, then *ConnectionLastUsed* is the last time that a packet was sent from eDirectory using this connection. Otherwise, it will be 0.

Table 84

Value	Description
TRUE	*ConnectionLastUsed* is the time that eDirectory last sent a packet on this connection.
FALSE	*ConnectionLastUsed* will be 0.

Sample NDS_LIMBER

Before eDirectory runs limber, it queries WAN Traffic Manager to see if this is an acceptable time for this activity. The traffic type NDS_LIMBER does not have a destination address; it requires a NO_ADDRESSES policy. If WAN Traffic Manager returns DONT_SEND, limber will be put off and rescheduled. The following variables are supplied.

◆ *Last* (Input Only, Type TIME)

The time of last limber since eDirectory started.

◆ *Version* (Input Only, Type INTEGER)

The version of eDirectory.

◆ *ExpirationInterval* (Output Only, Type INTEGER)

The expiration interval for all connections created while running limber checks.

Table 85

Value	Description
<0, 0	Use the default expiration interval (default).
>0	Expiration interval to be assigned to this connection.

◆ *CheckEachNewOpenConnection* (Output Only, Type INTEGER)

Table 86

Value	Description
0	Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default).
1	Call WAN Traffic Manager and let the policies decide whether to allow the connection.
2	Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail.

◆ *CheckEachAlreadyOpenConnection*(Output Only, Type INTEGER)

Table 87

Value	Description
0	Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default).
1	Call WAN Traffic Manager and let the policies decide whether to allow the connection.
2	Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail.

◆ *Next* (Output Only, Type TIME)

Time for the next round of limber checking. If this is not set, NDS_LIMBER will use the default.

Sample NDS_LIMBER_OPEN

NDS_LIMBER_OPEN is used only if either *CheckEachNewOpenConnection* or *CheckEachAlreadyOpenConnection* was set to 1 during the corresponding NDS_LIMBER query. This query is generated whenever *CheckEachNewOpenConnection* is 1 and eDirectory needs to open a new connection before running limber. This query is generated whenever *CheckEachNewOpenConnection* is 1 and eDirectory needs to open a new connection before doing schema synchronization or *CheckEachAlreadyOpenConnection* is 1 and eDirectory needs to reuse an already existing connection.

◆ *Version* (Input Only, Type INTEGER)

The version of eDirectory.

◆ *ExpirationInterval* (Input and Output, Type INTEGER)

The expiration interval that should be assigned to this connection.

Table 88

Value	Description
<0, 0	Use the default expiration interval (default).
>0	Expiration interval to be assigned to this connection.

◆ *ConnectionIsAlreadyOpen* (Input Only, BOOLEAN)

Value	Description
TRUE	eDirectory determines that it already has a connection to this address and can reuse that connection.
FALSE	eDirectory does not have a connection to this address and must create one.

◆ *ConnectionLastUsed* (Input Only, Type TIME)

If *ConnectionIsAlreadyOpen* is TRUE, then *ConnectionLastUsed* is the last time that a packet was sent from DS using this connection. Otherwise, it will be 0.

Value	Description
TRUE	*ConnectionLastUsed* is the time that eDirectory last sent a packet on this connection.
FALSE	*ConnectionLastUsed* will be 0.

Sample NDS_SCHEMA_SYNC

Before eDirectory synchronizes the schema, it queries WAN Traffic Manager to see if this is an acceptable time for this activity. The traffic type NDS_SCHEMA_SYNC does not have a destination address; it requires a NO_ADDRESSES policy. If WAN Traffic Manager returns DONT_SEND, schema synchronization will be put off and rescheduled. The following variables are supplied:

◆ *Last* (Input Only, Type TIME)

The time of the last successful schema synchronization to all servers.

◆ *Version* (Input Only, Type INTEGER)

The version of eDirectory.

◆ *ExpirationInterval* (Output Only, Type INTEGER)

The expiration interval for all connections created while synchronizing the schema.

Value	Description
<0, 0	Use the default expiration interval (default).
>0	Expiration interval to be assigned to this connection.

◆ *CheckEachNewOpenConnection* (Output Only, Type INTEGER)

Value	Description
0	Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default).
1	Call WAN Traffic Manager and let the policies decide whether to allow the connection.
2	Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail.

◆ *CheckEachAlreadyOpenConnection* (Output Only, Type INTEGER)

Value	Description
0	Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default).
1	Call WAN Traffic Manager and let the policies decide whether to allow the connection.
2	Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail.

Sample NDS_SCHEMA_SYNC_OPEN

NDS_SCHEMA_SYNC_OPEN is used only if either *CheckEachNewOpenConnection* or *CheckEachAlreadyOpenConnection* was set to 1 during the corresponding NDS_SCHEMA_SYNC query. This query is generated whenever *CheckEachNewOpenConnection* is 1 and eDirectory needs to open a new connection before doing schema synchronization or *CheckEachAlreadyOpenConnection* is 1 and eDirectory needs to reuse an already existing connection.

- *Version* (Input Only, Type INTEGER)

  The version of eDirectory.

- *ExpirationInterval* (Input and Output, INTEGER)

  The expiration interval that should be assigned to this connection.

Table 94

Value	Description
<0, 0	Use the default expiration interval (default).
>0	Expiration interval to be assigned to this connection.

- *ConnectionIsAlreadyOpen* (Input Only, BOOLEAN)

Table 95

Value	Description
TRUE	eDirectory determines that it already has a connection to this address and can reuse that connection.
FALSE	eDirectory does not have a connection to this address and must create one.

◆ *ConnectionLastUsed* (Input Only, Type TIME)

If *ConnectionIsAlreadyOpen* is TRUE, then *ConnectionLastUsed* is the last time that a packet was sent from eDirectory using this connection. Otherwise, it will be 0.

Table 96

Value	Description
TRUE	*ConnectionLastUsed* is the time that eDirectory last sent a packet on this connection.
FALSE	*ConnectionLastUsed* will be 0.

Sample NDS_SYNC

Whenever eDirectory needs to synchronize a replica, it makes a query to WAN Traffic Manager using the traffic type NDS_SYNC. The following variables are provided by eDirectory for use in WAN policies.

◆ *Last* (Input Only, Type TIME)

Time of the last successful synchronization to this replica.

◆ *Version* (Input Only, Type INTEGER)

The version of eDirectory.

◆ *ExpirationInterval* (Output Only, Type INTEGER)

The expiration interval for the connection to the server holding the updated replica.

Table 97

Value	Description
<0, 0	Use the default expiration interval (default).
>0	Expiration interval to be assigned to this connection.

ONOSPOOF.WMG

The policies in this group allow only existing WAN connections to be used. There are two policies:

 - Already Open, No Spoofing, NA

   This policy prevents the checking of backlinks, external references, and login restrictions, the running of janitor or limber, and schema synchronization except on existing WAN connections.

 - Already Open, No Spoofing

   This policy prevents all other traffic to existing WAN connections.

To prevent all traffic to existing connections, both policies must be applied.

OPNSPOOF.WMG

The policies in this group allow only existing WAN connections to be used but assumes that a connection that hasn't been used for 15 minutes is being spoofed and should not be used. There are two policies.

 - Already Open, Spoofing, NA

   This policy prevents the checking of backlinks, external references, and login restrictions, the running of janitor or limber, and schema synchronization except on existing WAN connections that have been open less than 15 minutes.

 - Already Open, Spoofing

   This policy prevents other traffic to existing WAN connections that have been open less than 15 minutes.

To prevent all traffic to existing connections open less than 15 minutes, both policies must be applied.

SAMEAREA.WMG

The policies in this group allow traffic only in the same network area. A network area is determined by the network section of an address. In a TCP/IP address, Wan Traffic Manager assumes a class C address (addresses whose first three sections are in the same network area). In an IPX address, all addresses with the same network portion are considered to be in the same network area. There are three policies.

◆ Same Network Area, NA

This policy prevents the checking of backlinks, external references, and login restrictions, the running of janitor or limber, and schema synchronization unless the traffic that would be generated is in the same network area.

◆ Same Network Area, TCPIP

This policy restricts TCP/IP traffic unless the traffic that would be generated is in the same TCP/IP network area.

◆ Same Network Area, IXP

This policy restricts IPX traffic unless that traffic that would be generated is in the same IPX network area.

TCPIP.WMG

The policies in this group allow only TCP/IP traffic. There are two policies.

◆ TCPIP, NA

This policy prevents the checking of backlinks, external references, and login restrictions, the running of janitor or limber, and schema synchronization unless the traffic that would be generated is TCP/IP.

◆ TCPIP

This policy prevents all other traffic unless the traffic is TCP/IP.

To prevent all non-TCP/IP traffic, both policies must be applied.

TIMECOST.WMG

The policies in this group restrict all traffic to between 1 a.m. and 1:30 a.m. but allows servers in the same location to talk continuously. This group uses the following policies, all of which must be applied:

◆ COSTLT20

This policy has a priority of 40 for NA and address traffic.

◆ Disallow Everything

This policy allows no traffic to be sent. If WAN Traffic Manager finds no (0) policies where the selector returned greater than 0, it defaults to SEND. This policy prevents this case.

- NDS Synchronization

  This policy restricts NDS_SYNC traffic to between 1 a.m. and 1:30 a.m.

- Start Rest. Procs, NA

  This policy allows all processes to start at any time, but WAN Traffic Manager must be consulted for each *_OPEN call. It schedules the process to run four times a day at 1:00, 7:00, 13:00, and 19:00.

- Start Unrest. Procs 1-1:30, NA

  This policy allows all processes to start between 1:00 a.m. and 1:30 a.m. and run to completion without further queries to WAN Traffic Manager. The processes run four times a day, every six hours. The 1:00 process is handled by this policy; the other processes are handled by the Start Rest. Procs, NA.

WAN Policy Structure

A WAN policy consists of three sections:

- "Declaration Section" on page 320
- "Selector Section" on page 324
- "Provider Section" on page 324

Declaration Section

The Declaration section of a policy contains definitions of local variables and variables coming in through a client request. These definitions are used within the Selector and Provider sections. These variables are stored along with system-defined variables.

Variable declarations are separated by a semicolon. Multiple declarations for the same type can be combined in one line or wrapped to the next line. They are not line-sensitive. A sample Declaration section is shown below:

```
REQUIRED INT R1;
REQUIRED TIME R2;
REQUIRED BOOLEAN R3,R4;
REQUIRED NETADDRESS R5,R6;
OPTIONAL INT P1 := 10;
OPTIONAL BOOLEAN := FALSE;
LOCAL INT L1 :=10;
LOCAL INT L2;
LOCAL TIME L3;
LOCAL BOOLEAN L4 :=TRUE, L5 :=FALSE;
LOCAL NETADDRESS L6;
```

The required and optional declarations are specific to a particular traffic type. Policies that do not contain the required variables will not run. The optional declarations must have a value to provide a default if none is passed in. WAN Traffic Manager provides system symbols (predefined variables) for use with all traffic types.

Each declaration consists of three parts:

  ◆ Scope

  ◆ Type

  ◆ List of names/optional value pairs

Scope

Valid scopes are listed in Table 98.

Table 98 **Valid Declaration Scopes**

Scope	Description
REQUIRED	Variables defined as REQUIRED in scope can be used in multiple sections, but only once within the Declaration section.
	No values can be defined for a REQUIRED scope variable. Its value must come from the GetWanPolicy request.

Scope	Description
OPTIONAL	Variables defined as OPTIONAL in scope can be used in multiple sections of a policy, but only once within the Declaration section.
	OPTIONAL scope variables are assigned to a default value. These values are not initialized. They are set only if a value is not passed. If a WAN policy request does not pass a new value to the parameter that matches in both name and type, the value defined in the Declaration is used when processing the policy.
	You must assign a value to variables defined as OPTIONAL in scope. Therefore, since TIME and NETADDRESS types cannot be initialized in the Declaration section, do not use an OPTIONAL scope with these variable types.
LOCAL	Variables defined as LOCAL in scope can be used in multiple sections, but only once within the Declaration section.
	LOCAL scope variables exist only for a particular policy, that is, their values are not returned to the calling client.
	All parameter types can be defined. However, since TIME and NETADDRESS types cannot be initialized in the Declaration section, do not assign values to these types.
SYSTEM	Variables defined as SYSTEM in scope can be used in multiple sections, but only once within the Declaration section.

Type

Valid types are listed in Table 99.

Table 99   Valid Declaration Types

Type	Description
INT	Reflects the traffic type of the GetWanPolicy request for which the policy is being run. For example, the following policy specifies a Traffic Type of NDS_SYNC: IF TrafficType=NDS_SYNC THEN *action* END.
BOOLEAN	Used for values of only TRUE or FALSE. The value will be indeterminate if it is not set in a Declaration or a WAN policy request.
TIME	TIME scope variables must receive their values in the Selector or Provider sections or from the WAN policy request. Do not assign values to TIME scope variables in the Declaration.
NETADDRESS	NETADDRESS scope variables must receive their values in the Selector or Provider sections. Do not assign values to NETADDRESS scope variables in the Declaration.

You cannot assign values to Time and Netaddress types in the Declaration section. If these types do not already have a value, they receive their values in the Selector or Provider sections. Only single types are initialized in the Declaration section.

Names/Optional Value Pairs

Variable names are combinations of alphanumeric characters in a string of any length. Since only the first 31 characters are used, a variable must begin with a unique 31-character string. A variable name must start with an alphabetic character, or the symbol is interpreted as a numeric constant.

Variable names are case-sensitive. For example, the variable *R1* is not the same as the variable *r1*. The underscore character (_) is allowed in variable names.

Values in a declaration must be constants rather than variables or expressions. Thus, the declaration LOCAL INT L2:= L3; is not allowed. A value initializing a variable in the Declaration section may be changed in the Selector and Provider sections of the policy.

Selector Section

The Selector section of a policy begins with the keyword SELECTOR and concludes with the keyword END. Selector sections are evaluated to determine which loaded policy will be used.

The Selector sections of all the currently-loaded policies are run to determine which policy has the greatest weight. When evaluated, the section returns a weight between 0-100, where 0 means do not use this policy, 1-99 means use this policy if no other policy returns a higher value, and 100 means use this policy.

The result of a Selector section is given in a RETURN declaration. If no RETURN declaration is made, a default value of 0 will be returned. The following is a sample Selector section:

```
SELECTOR
RETURN 49;
END
```

When the Selector sections of multiple policies are evaluated, more than one policy might return the same value. In this case, it is indeterminate which policy will be selected. All else being equal, a server policy will override a WAN policy.

For more information on writing declarations, see "Construction Used within Policy Sections" on page 325. See also "Provider Section" on page 324.

Provider Section

The Provider section begins with the keyword PROVIDER and concludes with the keyword END. The body of the Provider section consists of a list of declarations.

The result of this Declarations list should be a value representing the policy's suggestion to SEND or DONT_SEND.

The result of a Provider section is given in a RETURN declaration. If no RETURN declaration is made, a default value of SEND will be returned.

The following is a sample Provider section:

```
PROVIDER
RETURN SEND;
END
```

For more information on writing declarations, see "Construction Used within Policy Sections" on page 325.

Construction Used within Policy Sections

The following statements and constructions can be used, except as noted, in the Selector and Provider sections of a WAN policy. For more information on how to construct the Declaration section of a policy, see

Comments

Comments can be indicated by using /* at the beginning of the line and */ at the end, for example:

```
/* This is a comment. */
```

Comments can also be distinguished by // at the end of the line before a comment, for example:

```
IF L2 > L3 THEN //This is a comment.
```

IF-THEN Statement

IF-THEN statements are used to run a block of declarations conditionally.

Examples:

```
IF Boolean_expression THEN declarations
END

IF Boolean_expression THEN declarations
ELSE declarations
END

IF Boolean_expression THEN declarations
ELSIF Boolean_expression THEN declarations
END
```

IF *Boolean_Expression* THEN

This is the first clause in an IF-THEN statement. The Boolean expression is evaluated for a TRUE or FALSE result. If it is TRUE, the declarations that immediately follow are run. If it is FALSE, execution jumps to the next corresponding ELSE, ELSIF, or END declaration.

ELSE

This declaration marks the beginning of declarations that run if all corresponding preceding IF-THEN and ELSIF statements result in FALSE, for example:

```
IF Boolean_expression THEN statements
ELSIF Boolean_expression THEN statements
ELSIF Boolean_expression THEN statements
ELSE statements
END
```

ELSIF *Boolean_Expression* THEN

The Boolean expression is evaluated if the preceding IF-THEN declaration returns a FALSE. The ELSIF declaration is evaluated for a TRUE or FALSE result. If it is TRUE, the declarations that follow are run. If it is FALSE, execution jumps to the next corresponding ELSE, ELSIF, or END declaration, for example:

```
IF Boolean_expression THEN statements
ELSIF Boolean_expression THEN statements
ELSIF Boolean_expression THEN statements
END
```

END

The END declaration terminates an IF-THEN construction.

RETURN

The RETURN command gives the results of the Selector and Provider sections.

Selector

In a Selector section, the RETURN declaration provides the integer result used as a weight for the policy. RETURN assigns a policy weight between 0-100, where 0 means do not use this policy, 1-99 means use this policy if no other policy returns a higher value, and 100 means use this policy. If no RETURN declaration is made in a Selector section, a default value of 0 will be returned.

A semicolon is required to terminate the declaration, for example:

```
RETURN 49;
RETURN L2;
RETURN 39+7;
```

Provider

In a Provider section, the RETURN declaration provides the SEND or DONT_SEND result. If no RETURN declaration is made, a default value of SEND will be returned.

A semicolon is required to terminate the declaration, for example:

```
RETURN SEND;
RETURN DONT_SEND;
RETURN L1;
```

Assignment

The assignment declaration changes the value of a symbol using the := characters. The defined variable or system variable is stated first, then the := with a value, variable, or operation following. The assignment declaration must be terminated with a semicolon, for example:

```
variable.field:=expression; variable:=expression;
```

t1 and t2 are of type TIME, i1 and i2 are type INTEGER, and b1 and b2 are Boolean valid assignments:

```
t1 := t2;
b1 := t1 < t2;
i1 := t1.mday - 15;
b2 := t2.year < 2000
```

Invalid assignments:

```
b1 := 10 < i2 < 12;
```

(10 < i2) is Boolean, and a BOOLEAN cannot be compared to an INTEGER.

You could use b1 := (10 < i2) AND (i2 < 12); instead, for example:

```
  b2 := i1;
```

b2 is Boolean, and i1 is INTEGER. Therefore, they are incompatible types.

You could use b2 := i1 > 0; instead.

Strict type checking is performed. You are not allowed to assign an INT to a TIME variable.

Arithmetic Operators

You can include arithmetic operators in assignment declarations, RETURN declarations, or IF constructions. The valid operators are:

- Addition (+)
- Subtraction (-)
- Division (/)
- Multiplication (*)
- Module (MOD)

Use only INT variable types with arithmetic operators. Do not use TIME, NETADDRESS, and BOOLEAN variable types in arithmetic expressions.

Avoid operations that result in values outside of the range -2147483648 to +2147483648 or division by 0.

Relational Operators

You can use relational operators in IF constructions. The valid operators are:

- Equal to (=)
- Not equal to (< >)
- Greater than (>)
- Greater than or equal to (>=)
- Less than (<)
- Less than or equal to (<=)

You can use any relational operators with TIME and INT variable types. You can also use < > and = with NET ADDRESS and BOOLEAN variable types.

Logical Operators

The valid operators are:

- AND
- OR
- NOT
- Less than (<)
- Greater than (>)
- Equal to (=)

Bitwise Operators

You can use bitwise operators on INT variable types to return an integer value. The valid operators are:

- BITAND
- BITOR
- BITNOT

Complex Operations

The following precedence rules are enforced when processing complex expressions. Operators with the same precedence order are processed left-to-right. The order is as follows:

- Parenthesis
- Unary (+/-)
- BITNOT
- BITAND
- BITOR
- Multiplication, division, MOD
- Addition, subtraction
- Relational (>, >=, <, <=, =)
- NOT
- AND
- OR

If you are not certain of precedence, use parentheses. For example, if A, B, and C are integers or variables, A<B<C is not allowed. A<B would return a Boolean value, not an integer value, which cannot be compared to an integer C. However, (A<B) AND (B<C) would be syntactically correct.

PRINT

You can use PRINT declarations to send text and symbol values to the server's WAN Traffic Manager display screen and to the log file.

PRINT statements can have any number of arguments that can be literal strings, symbol names or members, integer values, or Boolean values, separated by commas.

You must enclose literal strings in double quotes (""). PRINT declarations must end in a semicolon (;), for example:

PRINT "INT=",10,"BOOL=",TRUE,"SYM=",R1;

TIME and NETADDRESS variables use formatted print declarations. TIME symbols are printed as follows:

```
m:d:y h:m
```

NETADDRESS variables are printed as follows:

```
Type length data
```

Type is either IP or IPX, *length* is the number of bytes, and *data* is the hexadecimal address string.

10 LDAP Services for Novell eDirectory

Lightweight Directory Access Protocol (LDAP) Services for Novell®
eDirectory™ is a server application that lets LDAP clients access information
stored in eDirectory. You can give different clients different levels of directory
access, and you can access the directory over a secure connection. These
security mechanisms let you make some types of directory information
available to the public, other types available to your organization, and certain
types available only to specified groups or individuals.

The directory features available to LDAP clients depend on the functionality
built in to the LDAP client and the LDAP server. For example, LDAP Services
for eDirectory lets LDAP clients read and write data in the eDirectory
database if the client has the necessary permissions. Some clients have the
capability to read and write data; others can only read directory data.

Some typical client features let clients do one or more of the following:

- Look up information about a specific person, such as an e-mail address or
  phone number.
- Look up information for all people with a given last name, or a last name
  that begins with a certain letter.
- Look up information about any eDirectory object or entry.
- Retrieve a name, e-mail address, business phone number, and home
  phone number.
- Retrieve a company name and city name.

The following sections provide information about LDAP services for eDirectory:

Understanding LDAP Services for eDirectory

Because of Internet and intranet technologies, networks are much larger and more complex than in the past. These larger networks create a greater need for a comprehensive directory service and a standard method for accessing information.

The Lightweight Directory Access Protocol (LDAP) is an Internet communications protocol that lets client applications access directory information. It is based on the X.500 Directory Access Protocol (DAP) but is less complex than a traditional client and can be used with any other directory service that follows the X.500 standard.

LDAP is used most often as the simplest directory access protocol.

For more information about LDAP, refer to the following Web sites:

- The University of Michigan (http://www.umich.edu/~dirsvcs/ldap/ldap.html)
- Innosoft's LDAP World (http://www.innosoft.com/ldapworld/)
- Directory Standards Demystified: LDAP Unlocks the Power of Your Network (http://www.novell.com/lead_stories/98/jul15/)
- LDAPzone.com (http://www.ldapzone.com)

Installing and Configuring LDAP Services for eDirectory

Novell LDAP Services for eDirectory is installed through the eDirectory installation. You can modify the default configuration of LDAP Services for eDirectory using ConsoleOne™. For more information, see Chapter 1, "Installing and Upgrading Novell eDirectory," on page 17.

Two new objects are added to your directory tree when eDirectory is installed:

- ◆ LDAP Server object. Use this object to set up and manage the Novell LDAP Server properties.

  See "Configuring the LDAP Server Object" on page 337 for more information.

- ◆ LDAP Group object. Use this object to set up and manage the way LDAP clients access and use the information on the Novell LDAP server.

  See "Configuring the LDAP Group Object" on page 338 for more information.

Loading and Unloading LDAP Services for eDirectory

LDAP Services for eDirectory can be loaded and unloaded manually. To load LDAP Services for eDirectory, enter the following commands:

Table 100   **Commands to Load LDAP Services for eDirectory**

Server	Command
NetWare®	At the console prompt, type **LOAD NLDAP.NLM**.
Windows* NT*/2000	In the DHOST (NDSCONS) screen select NLDAP.DLM > click Start.
Linux* or Solaris*	At the Linux or Solaris prompt, type **/usr/sbin/nldap -l**

To unload LDAP Services for eDirectory, enter the following commands:

Table 101   **Commands to Unload LDAP Services for eDirectory**

Server	Command
NetWare	At the console pompt, type `UNLOADNLDAP.NLM`.
Windows NT/2000	In the DHOST (NDSCONS) screen, select NLDAP.DLM > click Stop.
Linux or Solaris	At the Linux or Solaris prompt, type `/usr/sbin/nldap -u`

Tuning LDAP for eDirectory

The following are the optimal settings for eDirectory LDAP search and authentication on a server with two processors and 2 GB of RAM:

Table 102   **Optimal Settings for eDirectory LDAP Search and Authentication**

Maximum TCP port limit	45000
Maximum pending TCP connection requests	4096
Maximum packet receive buffers	10000
Minimum packet receive buffers	3000
Maximum physical receive packet sizes	2048
Maximum concurrent disk cache writes	2000
Maximum concurrent directory cache writes	500
Maximum directory cache buffers	200000
Maximum number of internal directory handles	100
Maximum number of directory handles	20

DSTRACE	!m*xxxxxx*
	Replace *xxxxxx* with the amount of RAM in bytes to use as cache.
	On NT, create a text file named _ndsdb.ini in the directory, then add this line.

Managing the Memory

eDirectory uses memory for the database cache and for directory usage. These are separate allocated memory pools. The directory engine uses memory from available memory pools in the operating system as needed. The database uses a cache pool that is defined by parameters detailed below. Usually, the more database cache given to eDirectory, the better the performance. However, since eDirectory uses available system memory for its buffers, if clients are performing queries that require large data sets to be returned, the size of the database cache might need to be decreased to have enough system memory for the directory to handle building the query responses.

The database engine uses the database cache to hold the most recently accessed blocks. This cache is initially defined with a fixed size of 16 MB. The size of this cache can be changed from the command line in shipping versions of eDirectory. The following example command will set the eDirectory database cache to 80 million bytes:

```
set dstrace=!mb 80000000
```

A file named _ndsdb.ini in the SYS:\NETWARE directory on a NetWare server, or in the directory containing the eDirectory database files on the Windows, Solaris, and Linux environments (normally \novell\nds\dbfiles) can also be defined. This text file simply needs to contain a line such as the following:

```
cache=80000000
```

Don't add any white space by the equals (=) sign

The cache in eDirectory 8.6 can be initialized with a hard limit just as with earlier versions. In addition, the upper and lower limits can be set either as hard numbers or as a percentage of available memory. Dynamic allocation control parameters allow the cache size to grow or shrink depending on use. If the proper configuration parameters are set, the database cache dynamically grows or shrinks based on other system resource needs.

Editing the _ndsdb.ini file can manually control database memory usage. The format for INI file commands is given below:

```
cache=cacheBytes # Set a hard memory limit
```

Alternative formats are shown in Table 103.

Table 103   **Alternative INI Commands**

Command	Description
cache=*cache_options*	Sets a hard limit or dynamically adjusting limit. Multiple cache options can be specified in any order, separated by commas. All are optional. They are as follows:
DYN or HARD	Dynamic or hard limit.
AVAIL or TOTAL	These only apply if a hard limit was chosen. Omit these options for a dynamic limit.
%:*percentage*	The percentage of available or total physical memory.
MIN:*bytes*	The minimum number of bytes.
MAX:*bytes*	The maximum number of bytes.
LEAVE:*bytes*	The minimum number of bytes to leave for the OS.
blockcachepercent=*percentage*	Splits the cache between the block and record cache.

If a hard limit is specified and the administrator wants to define the database cache to use a percentage of the memory, the administrator can select between a percentage of total memory or a percentage of available memory. Dynamic limits always refer to a percentage of available memory. The following command examples are all valid in the _ndsdb.ini file.

The following is an example dynamic limit of 75% available memory, a minimum of 16 million bytes, and 32 million bytes for the OS:

```
cache=DYN,%:75,MIN:16000000, LEAVE 32000000
```

The following is an example hard limit of 75% total physical memory, a minimum of 18 million bytes, and a maximum of 512 million bytes:

```
cache=HARD, TOTAL,%:75,MIN:18000000, MAX 512000000
```

The following is an example old style hard limit of 8 million bytes:

```
cache=8000000
```

The database cache is divided between block cache and record cache. Block cache holds data and index blocks that mirror the storage on the disk. Record cache holds in-memory representations of directory objects and attributes. If updating or adding to the directory, use the block cache setting. If performing mostly reads, use the record cache. It is possible to cause a thrashing condition in both caches if performing numerous sequential updates without allocating cache size properly. Unless specifically changed, the cache is allocated to be 50% block cache and 50% record cache. The blockcachepercent option can be included in the _ndsdb.ini file to specify the percentage of cache allocated to caching data and index blocks. (The default is 50%.) The remaining cache is used for entries.

For example, to designate 60% block cache and 40% record cache, enter the following:

```
blockcachepercent=60
```

Do not select 100% of the cache for either block or record cache and starve the other cache type. In general, do not allocate more than 75% of your cache memory to one or the other type.

Database cache settings can also be controlled using Novell iMonitor.

Although the cache size is dynamic depending on the amount of memory available, the DSTRACE command can still be used for custom environments.

Configuring the LDAP Server Object

The LDAP Server object stores configuration data for an LDAP Services for eDirectory server. During installation, an LDAP Server object named LDAP Server *server_name* is created (where *server_name* is the name of the server LDAP Services for eDirectory is installed on). The LDAP Server object is created in the same container as the Server object.

Each LDAP Server object configures one LDAP Services for eDirectory server. Do not assign the same LDAP Server object to more than one LDAP Services for eDirectory server. If you assign the LDAP Server object to another server, it is no longer assigned to the previous server.

1 In ConsoleOne, right-click the LDAP Server object > click Properties.

2 Enter the configurable parameters in the property pages.

   For more information on LDAP Server parameters, see the LDAP online help.

3 Click Apply > OK.

Configuring the LDAP Group Object

The LDAP Group object stores configuration data that can be applied to a single LDAP server or a group of LDAP servers. If you plan to implement the same configuration on multiple servers, configure one LDAP Group object and assign it to each of the LDAP Services for eDirectory servers from the LDAP Server General Page.

The LDAP Group configures the class and attribute mappings and security policies on the server. This greatly simplifies configuration changes, because one configuration change can be applied instantly to multiple LDAP servers.

During installation, an LDAP Group object named LDAP Group *server_name* is created in the same container as the Server object.

To configure the LDAP Group object, use ConsoleOne to complete the following steps:

1 In ConsoleOne, right-click the LDAP Group object > click Properties.

2 Enter the configurable parameters in the property pages.

   For more information on LDAP Group parameters, see the LDAP online help.

3 Click Apply > OK.

Configuring LDAP Server and LDAP Group Objects on Linux or Solaris Systems

You can use the LDAP configuration utility, ldapconfig, on Linux or Solaris systems to modify, view, and refresh the attributes of LDAP Server and Group objects.

Use the following syntax to view LDAP attribute values on Linux and Solaris systems:

```
ldapconfig [-t tree_name | -p host_name[:port]] [-w password]
[-a <user FDN>] [-V] [-R] [-H] [-f] -v attribute,attribute2...
```

Use the following syntax to modify values of LDAP attributes on Linux and Solaris systems:

```
ldapconfig [-t tree_name | -p host_name[:port]] [-w password]
[-a admin_FDN] -s attribute=value,...
```

Table 104   **ldapconfig Parameters**

ldapconfig Parameter	Description
-t	Name of the eDirectory tree where the component will be installed.
-p	Name of the host.
-w	Password of the user having administration rights.
-a	Fully distinguished name of the user having administration rights.
-v	Option to view the value of the LDAP attribute.
-s	Option to set a value for an attribute of the installed components.
-R	Refreshes the LDAP server.
-V	Lets you view the current LDAP configuration settings.
-H	Lets you view the usage and help strings.
-f	Allows operations on a filtered replica.

ldapconfig Parameter	Description
attribute	Configurable LDAP server or group attribute name. For more information, see "LDAP Server Attributes" on page 341 and "LDAP Group Attributes" on page 343.

Table 105 provides a description of configurable LDAP Server attributes:

Table 105   **LDAP Server Attributes**

LDAP Server Attribute	Description
LDAP Server	Fully Distinguished Name of the LDAP server object in eDirectory
LDAP Host Server	Fully Distinguished Name of the host eDirectory server that the LDAP server runs on.
LDAP Group	LDAP group object in eDirectory of which this LDAP server is a member.
LDAP Server Bind Limit	Number of clients that can simultaneously bind to the LDAP server. A value of 0 (zero) indicates no limit.
LDAP Server Idle Timeout	Period of inactivity from a client after which LDAP server will terminate connection with this client. A value of 0 (zero) indicates no limit.
LDAP Enable TCP	Indicates whether TCP (non-SSL) connections are enabled for this LDAP server. Range of values is 1 (yes) and 0 (no).
LDAP Enable SSL	Indicates whether SSL connections are enabled for this LDAP server. The range of values is 1 (yes) and 0 (no).
LDAP TCP Port	Port number on which LDAP server will listen for TCP (non SSL) connections.
LDAP SSL Port	Port number on which LDAP server will listen for SSL connections.
keyMaterialName	Name of the Certificate object in eDirectory which is associated with this LDAP server and which will be used for SSL LDAP connections.
searchSizeLimit	Maximum number of entries that the LDAP server will return to an LDAP client in response to a search. A value of 0 (zero) indicates no limit.
searchSizeLimit	Maximum number of seconds after which an LDAP search will be timed out by the LDAP server. A value of 0 (zero) indicates no limit.

LDAP Server Attribute	Description
extensionInfo	Extensions supported by the LDAP server.
filteredReplicaUsage	Specifies whether the LDAP server should use a filtered replica for an LDAP search. The range of values is 1 (use filtered replica) and 0 (do not use filtered replica).
sslEnableMutualAuthentication	Specifies whether SSL-based mutual authentication (Certificate-based client authentication) is enabled on the LDAP server
ldapEnablePSearch	Specifies whether or not the persistent search feature is enabled on the LDAP server. It can take the values, true or false.
ldapMaximumPSearchOperations	An integer value that limits the number of concurrent persistent search operations possible. A value of 0 specifies unlimited search operations.
ldapIgnorePSearchLimitsForEvents	Indicates whether size and time limits should be ignored after the persistent search request has sent the initial result set. It can take the values, true or false. If this is set to false, the entire persistent search operation is subject to the search limits. If either limit is reached the search will fail with the appropriate error message.

Table 106 provides a description of configurable LDAP Group attributes:

Table 106   **LDAP Group Attributes**

LDAP Group Attribute	Description
LDAP Server List	List of LDAP servers which are members of this group.
LDAP Allow Clear Text Password	Specifies whether the LDAP server allows transmission of passwords in clear text from an LDAP client. The range of values is 0 (no) and 1 (yes).
LDAP Search Referral Usage	Specifies how the LDAP server processes LDAP referrals. The range of values includes: ◆ Prefer Chaining (Chain Requests to Other NDS® Servers) The LDAP server will chain the request to other eDirectory servers rather than returning referrals, except when servicing a persistent search operation and an entry is not present on the local server, or when servicing any extended operation that returns referrals. ◆ Prefer Referrals The LDAP server will traverse the tree if there is no LDAP server running on another replica server that has the relevant objects. If there is an LDAP server running on another replica server, an LDAP referral of that server will be returned. ◆ Always Refer (All NDS LDAP Servers in Tree Must Support Referrals) The LDAP server will always return an LDAP referral. If no LDAP referral exists, an error is returned. ◆ Default Referral Shows the location of the default referral. An LDAP referral will be returned if the LDAP server cannot contact any other replica server in the same tree or if there is no other LDAP server running on the other replica server.

LDAP Services for Novell eDirectory   **343**

LDAP Group Attribute	Description
LDAP Referral	An LDAP referral will be returned if the LDAP server cannot contact any other replica server in the same tree or if there is no other LDAP server running on the other replica server. This is the default.

Examples

To view value of the attribute in the attribute list:

1 Enter the following command:

```
ldapconfig [-t tree_name | -p host_name[:port]] [-w
    password] [-a user_FDN] -v "LDAP Allow Clear Text
    Password","searchTimeLimit"
```

To configure the LDAP TCP port number:

1 Enter the following command:

```
ldapconfig [-t tree_name | -p host_name[:port]] [-w
    password] [-a admin_FDN] -s "LDAP TCP
    Port=389","searchSizeLimit=1000"
```

Understanding How LDAP Works with eDirectory

This section explains LDAP schema differences, class and attribute mappings, auxiliary class support, and LDAP syntax.

Connecting to eDirectory with LDAP

All LDAP clients bind or connect to eDirectory as one of the following types of users:

- [Public] User (Anonymous Bind)
- Proxy User (Proxy User Anonymous Bind)
- NDS User (NDS User Bind)

The type of bind the user authenticates with affects the content the LDAP client can access. LDAP clients access a directory by building a request and sending it to the directory. When an LDAP client sends a request through LDAP Services for eDirectory, eDirectory completes the request for only those attributes to which the LDAP client has the appropriate access rights.

For example, if the LDAP client requests an attribute value (which requires the Read right) and the user is granted only the Compare right to that attribute, the request is rejected.

Standard login restrictions and password restrictions still apply; however, any restrictions are relative to where LDAP is running. Time and address restrictions are honored, but address restrictions are relative to where the eDirectory login occurred—in this case, the LDAP server.

Connecting as a [Public] User

An anonymous bind is a connection that does not contain a username or password. If an LDAP client without a name and password binds to LDAP Services for eDirectory and the service is not configured to use a Proxy User, the user is authenticated to eDirectory as user [Public].

User [Public] is a nonauthenticated eDirectory user. By default, user [Public] is assigned the Browse right to the objects in the eDirectory tree. The default Browse right for user [Public] allows users to browse eDirectory objects but blocks user access to object attributes.

The default [Public] rights are typically too limited for most LDAP clients. Although you can change the [Public] rights, changing them will give these rights to all users. Because of this, we recommend that you use the Proxy User Anonymous Bind. For more information, see .

To give user [Public] access to object attributes, you must make user [Public] a trustee of the appropriate container or containers and assign the appropriate object and attribute rights.

Connecting as a Proxy User

A proxy user anonymous bind is an anonymous connection linked to an eDirectory username. If an LDAP client binds to LDAP for eDirectory anonymously, and the protocol is configured to use a Proxy User, then the user is authenticated to eDirectory as the Proxy User. The name is then configured in both LDAP Services for eDirectory and in eDirectory.

The anonymous bind traditionally occurs over port 389 in LDAP. However, you can manually configure different ports during the install to work on different nodes, such as Active Directory.

The key concepts of proxy user anonymous binds are as follows:

- All LDAP client access through anonymous binds is assigned through the Proxy User object.

- The Proxy User must have a null password or any password restrictions (such as password change intervals) because LDAP clients do not supply passwords during anonymous binds. Do not force the password to expire or allow the Proxy User to change passwords.

- You can limit the locations that the user can log in from by setting address restrictions for the Proxy User object.

- The Proxy User object must be created in eDirectory and assigned rights to the eDirectory objects you want to publish. The default user rights provide Read access to a limited set of objects and attributes. Assign the Proxy User Read and Search rights to all objects and attributes in each subtree where access is needed.

- The Proxy User object must be enabled on the General page of the LDAP Group object that configures LDAP Services for eDirectory. Because of this, there is only one Proxy User object for all servers in an LDAP group. For more information, see "Configuring the LDAP Group Object" on page 338.

- You can grant a Proxy User object rights to All Properties or Selected Properties. By default, the Proxy User receives rights to all properties.

To give the Proxy User rights to only selected properties:

1 Right-click the top container the Proxy User has rights over > click Add Trustees of This Objects.

2 Browse to Proxy User > click OK.

3 Deselect the following rights:

- Browse Entry Rights

- Read and Compare All Properties Rights

4 Click Selected Rights > select all inheritable rights for the Proxy User, such as mail stop and phone number.

To implement proxy user anonymous binds, you must create the Proxy User object in eDirectory and assign the appropriate rights to that user. Assign the Proxy User Read and Search rights to all objects and attributes in each subtree where access is needed. You also need to enable the Proxy User in LDAP Services for eDirectory by specifying the same proxy username.

1 In ConsoleOne, right-click the LDAP Group object.

2 Click Properties > General tab.

3 Enter the name of an eDirectory User object in the Proxy Username field.

Connecting as an eDirectory User

An eDirectory user bind is a connection that an LDAP client makes using a complete eDirectory username and password. The eDirectory user bind is authenticated in eDirectory, and the LDAP client is allowed access to any information the eDirectory user is allowed to access.

The key concepts of eDirectory user binds are as follows:

- eDirectory user binds are authenticated to eDirectory using the username and password entered at the LDAP client.

- The eDirectory username and password used for LDAP client access can also be used for NetWare client access to eDirectory.

- On non-SSL connections, the eDirectory password is transmitted in clear text on the path between the LDAP client and LDAP Services for eDirectory.

- If clear text passwords are not enabled, all eDirectory bind requests that include a username or password on non-SSL connections are rejected.

- If an eDirectory user password has expired, eDirectory bind requests for that user are rejected.

Allowing Clear Text Passwords

By default, eDirectory user bind requests using clear text (unencrypted) passwords are refused. Clear text passwords and eDirectory usernames entered by LDAP clients on non-SSL connections can be captured by network monitoring equipment. Anyone who captures an eDirectory username and password has immediate access to all the eDirectory objects that the captured username has access to. For this reason, eDirectory user binds are more secure on LDAP servers that are configured to use SSL.

To support eDirectory user binds on non-SSL connections, you must set the clear text passwords within the LDAP Group object.

1 In ConsoleOne, right-click the LDAP Group object.

2 Click Properties > the General tab.

3 Click Allow Clear Text Passwords.

Assigning eDirectory Rights for LDAP Clients

To assign eDirectory rights for LDAP clients:

1 Determine the type of username the LDAP clients will use to access eDirectory:

   ◆ [Public] User (Anonymous Bind)

   ◆ Proxy User (Proxy User Anonymous Bind)

   ◆ NDS User (NDS User Bind)

   See "Connecting to eDirectory with LDAP" on page 344 for more information.

2 If users will use one proxy user or multiple eDirectory usernames to access LDAP, use ConsoleOne to create these usernames in eDirectory or through LDAP.

3 Assign the appropriate eDirectory rights to the usernames that LDAP clients will use.

The default rights that most users receive provide limited rights to the user's own object. To provide access to other objects and their attributes, you must change the rights assigned in eDirectory.

When an LDAP client requests access to an eDirectory object and attribute, eDirectory accepts or rejects the request based on the LDAP client's eDirectory identity. The identity is set at bind time.

Class and Attribute Mappings

A class is a type of object in a directory, such as a user, server, or group. An attribute is a directory element that defines additional information about a specific object. For example, a User object attribute might be a user's surname or phone number. In ConsoleOne, classes are called object types or classes and attributes are called properties.

A schema is a set of rules that defines the classes and attributes allowed in a directory and the structure of a directory (where the classes can be in relation to one another). Because the schemas of the LDAP directory and the eDirectory directory are sometimes different, mapping LDAP classes and attributes to the appropriate eDirectory objects and attributes might be necessary. These mappings define the name conversion from the LDAP schema to the eDirectory schema.

LDAP Services for eDirectory provides default mappings. In many cases, the correspondence between the LDAP classes and attributes and the eDirectory object types and properties is logical and intuitive. However, depending on your implementation needs, you might want to reconfigure the class and attribute mapping.

In most instances, the LDAP class to eDirectory object type mapping is a one-to-one relationship. However, the LDAP schema supports alias names such as CN and common names that refer to the same attribute.

Mapping LDAP Group Attributes

The default LDAP Services for eDirectory configuration contains a predefined set of class and attribute mappings. These mappings map a subset of LDAP attributes to a subset of eDirectory attributes. If an attribute is not already mapped in the default configuration, an auto-generated map is assigned to the attribute. Also, if the schema name is a valid LDAP name with no spaces or colons, no mappings are required. You should examine the class and attribute mapping and reconfigure as needed.

1 In ConsoleOne, right-click the LDAP Group object.

2 Click the Attribute Map tab.

3 Add, delete, or modify the attributes you want.

Because there might be alternate names for certain LDAP attributes (such as CN and common name), you might need to map more than one LDAP attribute to a corresponding eDirectory attribute name. When LDAP Services for eDirectory returns LDAP attribute information, it returns the value of the first matched attribute it locates in the list.

If you map multiple LDAP attributes to a single eDirectory attribute, you should reorder the list to prioritize which attribute should take precedence because the order is significant.

Mapping LDAP Group Classes

When an LDAP client requests LDAP class information from the LDAP server, the server returns the corresponding eDirectory class information. The default LDAP Services for eDirectory configuration contains a predefined set of class and attribute mappings.

1 In ConsoleOne, right-click the LDAP Group object.

2 Click the Class Map tab.

3 Add, delete, or modify the classes you want.

LDAP Services for eDirectory is preconfigured to map a subset of LDAP classes and attributes to a subset of eDirectory classes and attributes.

The default LDAP Services for eDirectory configuration contains a predefined set of class and attribute mappings. These mappings map a subset of LDAP classes and attributes to a subset of eDirectory classes and attributes. If an attribute or class is not mapped in the default configuration, an auto-generated map is assigned to the attribute or class. Also, if the schema name is a valid LDAP name with no spaces or colons, no mappings are required. You should examine the class and attribute mapping and reconfigure as needed.

Auxiliary Classes

Refreshing the LDAP Server

Because the schemas of the LDAP directory and the eDirectory directory are different, mapping LDAP classes and attributes to the appropriate eDirectory objects and attributes is necessary. These mappings define the name conversion from the LDAP schema to the eDirectory schema.

No LDAP schema mappings are required for a schema entry if the name is a valid LDAP schema name. In LDAP, the only characters allowed in a schema name are alphanumeric characters and hyphens (-). No spaces are allowed in an LDAP schema name.

To ensure that searching by object IDs works after a schema extension other than LDAP, such as for .SCH files, you must refresh the LDAP server configuration if the schema is extended outside of LDAP.

1 In ConsoleOne, right-click the LDAP Server object.

2 Click Properties.

3 On the General tab, click Refresh LDAP Server Now.

Many-to-One Mappings

To support LDAP from eDirectory, LDAP Services uses mappings in the protocol level (instead of the directory service level) to translate between LDAP and eDirectory attributes and classes. Because of this, two LDAP classes or attributes can be mapped to the same eDirectory class or attribute.

For example, if you create a Cn through LDAP then search for attributeclass=CommonName, you can get back a Cn.

If you request all attributes, you get the attribute that is first in the mappings list for that class. If you ask for an attribute by name, you will get the correct name.

Table 107 on page 351 shows the many-to-one class mappings. Table 108 on page 351 shows the many-to-one attribute mappings.

Table 107   **Many-to-One LDAP Class Mappings**

LDAP Class Name	eDirectory Class Name
MailGroup	NSCP:mailGroup1
rfc822mailGroup	
GroupOfNames	Group
GroupOfUniqueNames	
Group	

Table 108   **Many-to-One LDAP Attribute Mappings**

LDAP Attribute Name	eDirectory Attribute Name
C	C
Country Name	
Cn	CN
CommonName	
Description	Description
MultiLineDescription	
L	L
Localityname	
Member	Member
uniqueMember	
o	O
organizationname	

LDAP Attribute Name	eDirectory Attribute Name
ou	OU
organizationalUnitName	
sn	Surname
surname	
st	S
stateOrProvinceName	
certificateRevocationList;binary	CertificateRevocationList
certificateRevocationList	
authorityRevocationList;binary	AuthorityRevocationList
authorityRevocationList	
deltaRevocationList;binary	DeltaRevocationList
deltaRevocationList	
cACertificate;binary	CACertificate
cACertificate	
crossCertificatePair;binary	CrossCertificatePair
crossCertificatePair	
userCertificate;binary	UserCertificate
userCertificate	

Enabling Non-Standard Schema Output

eDirectory contains a compatibility mode switch that allows non-standard schema output so that current ADSI and old Netscape* clients can read the schema. This is implemented by setting an attribute in the LDAP Server object. The attribute name is nonStdClientSchemaCompatMode. The LDAP Server object is usually in the same container as the Server object.

The non-standard output does not conform to the current IETF standards for LDAP, but it will work with the current version of ADSI and old Netscape clients.

In non-standard output format:

- SYNTAX OID is single quoted.

- No upper bounds are output.

- No X- options are output.

- If more than one name is present, only the first encountered is output.

- Attributes or classes without an OID defined are output as attributename-oid or classname-oid in lowercase.

- Attributes or classes with a dash in the name and no defined OID are not output.

To enable non-standard schema output:

1 In ConsoleOne, right-click the LDAP Server object.

2 Click Properties > the General tab.

3 Click Enable Non-Standard Client Schema Compatible Mode > Refresh NLDAP Server Now.

4 Click Apply > OK.

You can also add and set this attribute using LDAP Modify calls. If this is done through LDAP you need to refresh your LDAP server. For more information, see "Refreshing the LDAP Server" on page 350.

Specialized LDAP Schema Files

The following specialized LDAP schema files are available from the Novell download site (http://www.novell.com/download):

- inetOrgPerson

The default LDAP schema maps the object class inetOrgPerson to the eDirectory User class. Since this is a direct mapping and not a schema extension, the attributes of User are applied to inetOrgPerson.

The Novell eDirectory download site contains the NOV_INET.ZIP file. This file contains a separate schema extension file (NOV_INET.SCH) and instructions (NOV_INET.TXT) that modify the eDirectory User class to provide all the attributes in compliance with the definition of the informational RFC 2798. Adding this schema extension exposes an object class with all the RFC and Netscape attributes specified by the IETF (http://www.ietf.org/rfc/rfc2798.txt?number=2798).

◆ residentialPerson

The default schema file shipping with this release does not provide an object class definition for residentialPerson. The eDirectory download site contains the RPERSON.ZIP file. The file contains the schema extension file (RPERSON.SCH) and an instruction file (RPERSON.TXT).

If you plan to use this object class, we recommend that you extend the schema instead of simply mapping residentialPerson to the eDirectory User class.

◆ newPilotPerson

The default schema file shipping with this release does not provide an object class definition for newPilotPerson. The eDirectory download site contains the NPERSON.ZIP file. The file contains the schema extension file (NPERSON.SCH) and an instruction file (NPERSON.TXT).

If you plan to use this object class, you should extend the schema instead of simply mapping newPilotPerson to User in eDirectory.

◆ photo

If you try to extend the schema to include a photo attribute, this attribute might conflict with a prior definition for this class. The photo attribute can be located in either the INETORGPERSON.ZIP file or the NOV_INET.ZIP schema extension file described above. The photo attribute can be defined either as a SYN_STREAM (which can only be single-valued in eDirectory), or as a SYN_OCTET_STRING (which can be multivalued).

RFC 1274 calls for a photo to be multivalued with a maximum string length of 250,000 octets. eDirectory allows a maximum of 63,000 octets in a SYN_OCTET_STRING. You will have to select the photo restrictions that you prefer.

The schema extension file for inetOrgPerson contains an ldapPhoto attribute definition based on multivalue and SYN_OCTET_STRING.

Applying Schema Files on NetWare

1 Copy the .SCH file to the SYS:SYSTEM\SCHEMA directory.

2 Run NWCONFIG.NLM from your server console.

3 Select Directory Options > Extend Schema.

4 Log in with your administrator name and password.

5 Press F3 to specify a different path.

6 Enter **`SYS:SYSTEM\SCHEMA\`** and the name of the .SCH file.

7 In ConsoleOne, right-click the LDAP Server object.

8 Click Properties > Refresh LDAP Server Now.

Applying Schema Files on NT

1 Load INSTALL.DLM.

2 Select Install Additional Schema Files.

3 Log in with your administrator name and password, then select a schema file.

eDirectory also provides LDAP LDIF schema extension support. For more information, see "Using LDIF to Extend the Schema" on page 492.

Applying Schema on Linux or Solaris

You can use the ndssch utility to apply schema on Linux or Solaris systems. For more information, see "Using the ndssch Utility to Extend the Schema on Linux or Solaris" on page 157.

Syntax Differences

LDAP and eDirectory use different syntaxes. Some important differences are:

- "Commas" on page 356
- "Typeful Names Only" on page 356
- "Escape Character" on page 356
- "Multiple Naming Attributes" on page 356

Commas

LDAP uses commas as delimiters rather than periods. For example, a distinguished (or complete) name in eDirectory looks like this:

CN=JANEB.OU=MKTG.O=EMA

Using LDAP syntax, the same distinguished name would be:

CN=JANEB,OU=MKTG,O=EMA

Some additional examples of LDAP distinguished names include:

CN=Bill Williams,OU=PR,O=Bella Notte Corp
CN=Susan Jones,OU=Humanities,O=University College London,C=GB

Typeful Names Only

eDirectory uses both typeless (.JOHN.MARKETING.ABCCORP) and typeful (CN=JOHN.OU=MARKETING.O=ABCCORP) names. LDAP uses only typeful names with commas as the delimiters (CN=JOHN,OU=MARKETING,O=ABCCORP).

Escape Character

The backslash (\) is used in LDAP distinguished names as an escape character. If you use the plus sign (+) or the comma (,), you can escape them with a single backslash character. Some examples include

CN=Pralines\+Cream,OU=Flavors,O=MFG (CN is Pralines+Cream)

CN=D. Cardinal,O=Lionel\,Turner and Kaye,C=US (O is Lionel, Turner and Kaye)

See Internet Engineering Task Force RFC 232 (http://www.ietf.org/rfc/rfc2253.txt?number=2253) for more information.

Multiple Naming Attributes

Objects can be defined with multiple naming attributes in the schema. In both LDAP and eDirectory, the User object has two: CN and OU. The plus sign (+) separates the naming attributes in the distinguished name. If the attributes are not explicitly labeled, the schema determines which string goes with which attribute (the first would be CN, the second is OU for eDirectory and LDAP). You can reorder them in a distinguished name if you manually label each portion.

For example, the following are two relative distinguished names:

Smith (CN is Smith CN=Smith)

Smith+Lisa (CN is Smith, the OU is Lisa CN=Smith OU=Lisa)

Both relative distinguished names (Smith and Smith+Lisa) can exist in the same context because they must be referenced by two completely different relative distinguished names.

Supported Novell LDAP Controls and Extensions

The LDAP 3 protocol allows LDAP clients and LDAP servers to use controls and extensions for extending an LDAP operation. Controls and extensions allow you to specify additional information as part of a request or a response. Each extended operation is identified by an OID. LDAP clients can send extended operation requests specifying the OID of the extended operation that should be performed and the data specific to that extended operation. When the LDAP server receives the request, it performs the extended operation and sends a response containing an OID and any additional data to the client.

For example, a client can include a control that specifies a sort with the search request that it sends to the server. When the server receives the search request it sorts the search results before sending the search results back to the client. Servers can also send controls to clients. For example, a server can send a control with the authentication request that informs the client about password expiration.

By default the eDirectory LDAP server will load all system extensions and selected optional extensions and controls when the LDAP server starts up. The extensionInfo attribute of LDAP Server object for optional extensions allows the system administrator to select or deselect the optional extensions and controls.

To enable extended operations, LDAP 3 protocol requires servers to provide a list of supported controls and extensions in the supportedControl attribute and supportedExtension attribute in the root DSE. Root DSE (DSA [Directory System Agent] Specific Entry) is an entry that is located at the root of the Directory Information Tree (DIT).

Table 109 lists the supported LDAP extensions:

Table 109   **Supported LDAP Extensions**

LDAP Extension	Extension Type	Description
Refresh LDAP Server	System	Allows the LDAP Server to restart after rereading its configuration from the DS.
LBURP	Optional	LDAP Bulk Update/Replication Protocol (LBURP). The eDirectory Import/Export utility maximizes network and eDirectory server processing efficiency by using LBURP to transfer data to the server. Using LBURP during an LDIF import considerably improves the speed of LDIF import.
libldapxs	Optional	Converts eDirectory domain names to LDAP domain names and vice-versa.
LDAP Partitioning	Optional	Includes replica operations such as add replica, remove replica, change replica information, get replica information, list replicas and so on.
Identity Management	Optional	Includes context naming and management

Table 110 lists the supported LDAP controls:

Table 110   **Supported LDAP Controls**

LDAP Control	Description
Virtual List View	When you send a search request with this control along with a server-side sorting control to the server, the server sorts the results and returns the specified subset of entries back to your client. The request OID of this control is 1.2.840.113556.3.4.9 OID. The response OID of this control is 1.2.840.113556.3.4.10.
Server Side Sort	When you send a search request with this control to the server, the server sorts the results before sending them back to your client. The request OID of this control is 1.2.840.113556.1.4.473. The response OID for this control is 1.2.840.113556.1.4.474.
Persistent Search	When you send a search request with this control, the server returns the results to the client, but retains the connection with the client. The server notifies the client whenever there is any change to the entries on the local server that match the specified search filter. The request OID of this control is 2.16.840.1.113730.3.4.3. The response OID for this control is 2.16.840.1.113730.3.4.7.

Enabling Secure LDAP Connections

A trusted root provides the basis of trust in a public key infrastructure. A trusted root is a certificate that you implicitly trust and that you install into your browser (or other client software). In the context of SSL security, your browser automatically validates any server certificate that was signed by one of the trusted roots that has been installed and activated in your browser. In eDirectory, the CA and Key Material objects are installed by default when you accept the certificate server. Netscape and Microsoft Internet Explorer browsers are preconfigured with various trusted root certificates.

Understanding the Secure Sockets Layer Protocol

LDAP Services for eDirectory supports the SSL protocol to ensure that the connection that data is transmitted over is secure and private.

SSL establishes and maintains secure communication between SSL-enabled servers and clients across the Internet. To ensure message integrity, SSL uses a hashing algorithm. To ensure message privacy, SSL provides for the creation and use of encrypted communications channels. To prevent message forgery, SSL allows the server and, optionally, the client to authenticate each other during the establishment of the secure connection.

Key Material Object

To implement the authentication and encryption processes, SSL uses a cryptographic mechanism called public keys. To establish a secure connection, the server and the client exchange their public keys to establish a session key. The session key encrypts the data for the life of the connection. A subsequent LDAP connection over SSL will result in the generation of a new session key that is different from the previous one.

Each LDAP server requires a digital certificate to implement SSL. Digital certificates are issued by a certification authority (CA). Certificates are stored in the Key Material object. You can use the ConsoleOne Novell Certificate Server snap-in to request, manage, and store certificates in eDirectory. Refer to the Novell Certificate Server™ help for more information on setting up a certificate on a server. (Click Help on any Key Material object page.)

In order for the LDAP server to use a specific certificate for LDAP SSL connectivity once it is stored in eDirectory, you must indicate the Key Material object containing the certificate on the LDAP Server SSL Configuration Page in ConsoleOne.

1 Right-click the LDAP Server object.

2 Click the SSL Configuration tab.

3 Enter the name of the Key Material object in the SSL Certificate field.

Multiple Key Material objects can hold various SSL certificates within eDirectory, but the LDAP server uses only the one defined by this parameter for SSL connections. You can enter the partial name of the Key Material object or browse through a list of available objects.

Configuring SSL

SSL can be configured on both the client and server to ensure the identity of both parties, but clients do not require digital certificates to communicate securely. As the LDAP server listens for SSL connections on a special port, the client can initiate the connection over that port automatically when accepting the certificate server or manually by performing the following:

1 In ConsoleOne, right-click the LDAP Server object.

2 Click the SSL Configuration tab.

3 Enter the SSL port number for the LDAP services on an eDirectory server.

   You can also click Disable SSL Port so that encrypted messages cannot be exchanged through the network.

When you make changes to your LDAP Services for eDirectory configuration using ConsoleOne, some of the changes take effect dynamically without restarting the LDAP server.

Enabling Mutual Authentication

To prevent message forgery, SSL lets the server and optionally the client authenticate to each other during the establishment of the secure connection.

1 In ConsoleOne, right-click the LDAP Server object.

2 Click the SSL Configuration tab.

3 Select Enable Mutual Authentication.

Exporting the Trusted Root

You can export the trusted root automatically when accepting the certificate server or manually by performing the following:

1 Export the self-assigned CA from eDirectory.

   1a Right-click the Key Material Object > click Properties.

   1b Click the Certificates tab > Public Key Certificates.

   1c Click Export.
   Change file extension to .x509 if the file is going to be imported into a Netscape browser.

2 Install the self-assigned CA in all browsers that establish secure LDAP connections to eDirectory.

Internet Explorer 5 exports root certificates automatically with a registry update. The traditional .x509 extension used by Microsoft is required. See

Importing the Trusted Root into the Browser

Importing the Trusted Root into Netscape Navigator

1 Click File > Open Page.

2 Click Choose File > open the trusted root file that was previously exported.

This launches the New Certificate Authority Wizard.

The New Certificate Authority Wizard does not launch if you do not have the correct file extension registered on your workstation. This is usually the case if you have installed Internet Explorer 5 or Windows NT Service Pack 4 or later.

To fix this problem:

- ◆ Exit Navigator.

- ◆ Run the file X509.REG (located in *install_directory*\NDS, where *install_directory* is the directory name you selected when you installed eDirectory).

- ◆ Rename the trusted root certificate file you exported to the .X509 extension.

- ◆ Import the certificate into Navigator.

3 Follow the online prompts.

4 Check Accept This Certificate Authority for Certifying Network Sites.

Importing the Trusted Root into Internet Explorer

1 Select File > Open.

2 Locate and select the trusted root file that was previously exported.

This launches the New Site Certificate Wizard.

3 Follow the online prompts.

Internet Explorer 5 imports root certificates automatically.

Using LDAP Tools on Linux or Solaris

eDirectory includes LDAP tools that help you manage the LDAP directory sever. The following sections provide information about using LDAP tools for eDirectory:

- "Modifying Entries in the LDAP Directory Server" on page 363
- "Modifying the Relative Distinguished Name of Entries in LDAP Directory Server" on page 365
- "Deleting Entries from the LDAP Directory Server" on page 366
- "Searching Entries in the LDAP Directory Server" on page 368

To perform secure LDAP tools operations, refer to "Ensuring Secure eDirectory Operations on Linux and Solaris Systems" on page 77, and include the DER file in all command line LDAP operations that establish secure LDAP connections to eDirectory.

Modifying Entries in the LDAP Directory Server

The ldapmodify tool opens a connection to an LDAP server, binds, and modifies or adds entries. The entry information is read from standard input or from file using the -f option.

Use the following syntax to perform ldapmodify operations:

```
ldapmodify [-a] [-b] [-c] [-r] [-n] [-v] [-d debuglevel] [-e
key filename] [-D binddn] [[-W]|[-w passwd]] [-h ldaphost] [-
p ldap-port] [-f file]
```

Table 111   ldapmodify Parameters

ldapmodify Parameter	Description
-a	Adds new entries. The default for ldapmodify is to modify existing entries.
-b	Assumes that any values that start with a slash (/) are binary values and that the actual value is in a file whose path is specified in the place where values normally appear.
-c	Continuous operation mode. Errors are reported, but ldapmodify will continue with modifications. The default is to exit after reporting an error.

ldapmodify Parameter	Description
-r	Replaces existing values by default.
-n	Displays what would be executed, but does not actually modify entries. Useful for debugging in conjunction with -v.
-v	Uses verbose mode, with many diagnostics written to standard output.
-F	Forces application of all changes regardless of the contents of input lines that begin with replica: (by default, replica: lines are compared against the LDAP server host and port in use to decide if a replog record should actually be applied).
-d debuglevel	Sets the LDAP debugging level to debuglevel. ldapmodify must be compiled with LDAP_DEBUG defined for this option to have any effect.
-e	File certificate file name for SSL bind.
-f *file*	Reads the entry modification information from file instead of from standard input.
-D *binddn*	Binds to the X.500 directory. binddn should be a string-represented DN as defined in RFC 1779.
-W *prompt_for_simple_ authentication*	Use instead of specifying the password on the command line.
-w passwd	Use passwd as the password for simple authentication.
-h *ldaphost*	Specifies an alternate host on which the LDAP server is running.
-p *ldapport*	Specifies an alternate TCP port where the LDAP server is listening.

Example

For modifying entries in the LDAP directory server, enter the following:

```
ldapmodify -h xyzcompany.com  -D cn=admin,o=xyzcompany -w
  treasure -f T01.mod
```

Modifying the Relative Distinguished Name of Entries in LDAP Directory Server

You can use ldapmodrdn to modify the relative distinguished name (RDN) of entries in the LDAP directory server. The ldapmodrdn tool opens a connection to an LDAP server, binds, and modifies the RDN of entries. The entry information is read from standard input, from file using the -f option, or from the command line pair dn and rdn.

Use the following syntax to perform ldapmodrdn operations:

```
ldapmodrdn [-r] [-n] [-v] [-c] [-d debuglevel] [-e key
filename] [-D binddn] [[-W]|[-w passwd]] [-h ldaphost] [-p
ldapport] [-f file] [dn rdn]
```

Table 112   ldapmodrdn Parameters

ldapmodrdn Parameter	Description
-r	Removes old RDN values from the entry. The default is to retain old values.
-n	Displays what would be executed, but does not actually change entries. Useful for debugging in conjunction with -v.
-v	Uses verbose mode, with many diagnostics written to standard output.
-c	Continuous operation mode. Errors are reported, but ldapmodify will continue with modifications. The default is to exit after reporting an error.
-d debuglevel	Sets the LDAP debugging level to debuglevel. ldapmodrdn must be compiled with LDAP_DEBUG defined for this option to have any effect.
-e	File certificate file name for SSL bind.
-f *file*	Reads the entry modification information from file instead of from standard input or the command line.
-D *binddn*	Binds to the X.500 directory. binddn should be a string-represented DN as defined in RFC 1779.
-w	Prompts for simple authentication. Used instead of specifying the password on the command line.

ldapmodrdn Parameter	Description
-w passwd	Use passwd as the password for simple authentication.
-h *ldaphost*	Specifies an alternate host on which the LDAP server is running.
-p	Specifies an alternate TCP port where the LDAP server is listening.

Example

For modifying the RDN of entries in LDAP directory server, enter the following:

```
ldapmodrdn -r -D cn=admin,o=xyzcompany -w treasure
  cn=UserDetail,o=xyzcompany  cn=UserInfo
```

Deleting Entries from the LDAP Directory Server

You can use ldapdelete to delete entries from the LDAP directory server. The ldapdelete tool opens a connection to an LDAP server, binds, and deletes one or more entries. If one or more dn arguments are provided, entries with those distinguished names are deleted. Each dn should be a string-represented DN as defined in RFC 1779. If no dn arguments are provided, a list of DNs is read from the standard input or from file, if the -f flag is used.

Use the following syntax to perform ldapdelete operations:

```
ldapdelete [-n] [-v] [-c] [-d debuglevel] [-e key filename]
[-f file] [-D binddn] [[-W]| [-w passwd]] [-h ldaphost] [-p
ldapport] [dn]...
```

Table 113   ldapdelete Parameters

ldapdelete Parameter	Description
-n	Displays what would be executed, but does not actually delete entries. Useful for debugging in conjunction with -v.
-v	Uses verbose mode, with many diagnostics written to standard output.
-c	Continuous operation mode. Errors are reported, but ldapdelete will continue with deletions. The default is to exit after reporting an error.
-d debuglevel	Sets the LDAP debugging level to debuglevel. ldapdelete must be compiled with LDAP_DEBUG defined for this option to have any effect.
-e	File certificate file name for SSL bind.
-f *file*	Reads a series of lines from file, performing one LDAP search for each line. In this case, the filter given on the command line is treated as a pattern where the first occurrence of% is replaced with a line from file.
-D *binddn*	Binds to the X.500 directory. binddn should be a string-represented DN as defined in RFC 1779.
-W	Prompts for simple authentication. Used instead of specifying the password on the command line.
-w passwd	Use passwd as the password for simple authentication.
-h *ldaphost*	Specifies an alternate host on which the LDAP server is running.
-p *ldapport*	Specifies an alternate TCP port where the LDAP server is listening.

Example

To delete entries from the LDAP directory server, enter the following:

```
ldapdelete -D cn=admin,o=xyzcompany -w treasure -f T01.del
```

LDAP Services for Novell eDirectory   **367**

Searching Entries in the LDAP Directory Server

You can use ldapsearch to search entries in the LDAP directory server. The ldapsearch tool opens a connection to an LDAP server, binds, and performs a search using the specified filter. The filter should conform to the string representation for LDAP filters as defined in RFC 2254. If ldapsearch finds one or more entries, the attributes specified by the *attrs* parameter are retrieved and the entries and values are printed to standard output. If no value is specified for this parameter, all attributes are returned.

Use the following syntax to perform ldapsearch operations:

```
ldapsearch [-n] [-u] [-v] [-t] [-A] [-C] [-L] [-R] [-d
debuglevel] [-e key filename] [-F sep] [-f file] [-D binddn]
[[-W]| [-w bindpasswd]] [-h ldaphost] [-p ldapport] [-b
searchbase] [-s scope] [-a deref] [-l time limit] [-z size
limit] filter [attrs....]
```

Table 114   **ldapsearch Parameters**

ldapsearch Parameter	Description
-n	Displays what would be executed, but does not actually perform the search. Useful for debugging in conjunction with -v.
-u	Includes the user-friendly form of the Distinguished Name (DN) in the output.
-v	Runs in verbose mode, with many diagnostics written to standard output.
-t	Writes retrieved values to a set of temporary files. This is useful for dealing with non-ASCII values such as jpegPhoto or audio.
-A	Retrieves attributes only (no values). This is useful when you only want to verify if an attribute is present in an entry and not specific values of the attribute.
-C	Chase referrals automatically.
-L	Displays search results in LDIF format. This option also turns on the -B option, and causes the -F option to be ignored.

ldapsearch Parameter	Description
-R	Does not automatically follow referrals returned while searching. ldapsearch must be compiled with LDAP_REFERRALS defined for referrals to be automatically followed by default, and for this option to have any effect.
-e	File certificate filename for SSL bind.
-F sep	Uses sep as the field separator between attribute names and values. The default separator is =, unless the -L flag has been specified, in which case this option is ignored.
-S attribute	Sorts the entries returned based on attribute. The default is not to sort entries returned. If the attribute is a zero-length string (""), the entries are sorted by the components of their distinguished names. Note that ldapsearch normally prints out entries as it receives them. The use of the -S option causes all entries to be retrieved, sorted, and then printed.
-d debuglevel	Sets the LDAP debugging level to debuglevel. ldapsearch must be compiled with LDAP_DEBUG defined for this option to have any effect.
-f *file*	Reads a series of lines from file, performing one LDAP search for each line. In this case, the filter given on the command line is treated as a pattern where the first occurrence of% is replaced with a line from file. If file is a single - character, then the lines are read from standard input.
-D *binddn*	Binds to the X.500 directory. binddn should be a string-represented DN as defined in RFC 1779.
-W *prompt_for_simple_ authentication*	Use instead of specifying the password on the command line.
-w bindpasswd	Use bindpasswd as the password for simple authentication.
-h *ldaphost*	Specifies an alternate host on which the LDAP server is running.
-p *ldapport*	Specifies an alternate TCport where the LDAP server is listening.

ldapsearch Parameter	Description
-b searchbase	Use as the starting point for the search instead of the default.
-s *scope*	Specifies the scope of the search. The scope should be base, one, or sub to specify a base object, one-level, or subtree search. The default is sub.
-a *deref*	Specifies how aliases dereferencing is done. The values for this parameter can be one or never, always, search, or find to specify that aliases are never dereferenced, always dereferenced, dereferenced when searching, or dereferenced only when locating the base object for the search. The default is to never dereference aliases.
-l *time_limit*	Waits at most *timelimit* seconds for a search to complete.
-a *size_limit*	Waits at most *sizelimit* seconds for a search to complete.

Example

For searching entries in the LDAP directory server, enter the following:

```
ldapsearch -h xyzcompany.com -b o=xyzcompany -p 636 -e
  TrustedRootCert -D cn=admin,o=xyzcompany -w treasure
  cn=admin
```

Persistent Search

Persistent search allows you to track changes to a set of entries on an LDAP server that match a specified search criteria. After the initial search is performed, the server keeps track of the search criteria and sends back information when any entry that matches the criteria is added, modified, deleted, or renamed.

Persistent search alters the standard LDAP search operation so that it does not end after the initial set of entries matching the search criteria is returned. Instead, LD8klju8AP servers keep the search operation going until it is abandoned by the client or until the client unbinds. This provides clients and servers participating in persistent search with an active channel through which

entries that change (and additional information about the changes that occur) can be communicated.

NOTE: A persistent search operation may not retrieve the updated information, if entries in the eDirectory database are updated frequently.

This section provides the following information:

Benefits of Persistent Search

Persistent search provides the following benefits:

Cache Consistency

An LDAP client application with high performance needs might want to maintain a temporary, local cache of information obtained through LDAP search, compare, or bind operations. To improve performance, the local cache is always consulted before sending a request to an LDAP server. A persistent search request where the changesOnly flag is FALSE can be used if it is desirable to prime the cache; otherwise changesOnly would typically be set to TRUE in the request.

Caches are used for reasons other than performance improvement as well. In some cases, they arise naturally out of a particular application's design. For example, an LDAP client designed for administration of information held in LDAP servers will undoubtedly generate screen displays that show information gleaned from an LDAP server. The screen display is a cache that is active and visible until the user of the application takes some action that causes different information to be displayed. A refresh button or similar control may be provided to the user to allow them to update the cached display. A persistent search request can be used instead by the administrative application to automatically refresh the screen display as soon as the underlying LDAP information changes.

Synchronization

Some LDAP clients, such as those that execute on a portable computer, may maintain a partial or complete offline copy of the entries stored in an LDAP server. While connected to the network, such a client can direct all queries to the copy of data it holds and use a persistent search to actively maintain the contents of the offline copy (alternatively, the client could direct requests to the LDAP server that is the source of the data).

Triggered Actions

An LDAP client application may want to take some action when an entry in the directory is changed. A persistent search request can be used to proactively monitor one or more LDAP servers for interesting changes that in turn cause specific actions to be taken by an application. For example, an electronic mail repository may want to perform a "create mailbox" task when a new person entry is added to an LDAP directory and a "delete mailbox" task when a person entry is deleted from an LDAP directory.

Enabling and Configuring Persistent Search

1 In ConsoleOne, right-click the LDAP-Server object.

2 Click the Persistent Search tab.

3 To enable or disable persistent search, click Enable Persistent Search.

4 Specify the maximum number of persistent search operations that can be running at the same time.

   Enter 0 to activate unlimited search operations.

5 Specify whether size and time limits should be ignored after the persistent search request has sent the initial search result set.

   If this option is not selected, the entire persistent search operation is subject to the search restrictions. If either limit is reached, the search will fail with the appropriate error message.

6 Click OK.

11 Implementing the Service Location Protocol

The Service Location Protocol (SLP) is an Internet standard protocol (RFC 2165) that enables client applications to dynamically discover services in TCP/IP networks. Novell® provides implementations of SLP for NetWare®, Windows* 95, Windows 98, Windows NT*, and Windows 2000.

Understanding SLP Components

SLP defines three types of agents:

- User Agents
- Service Agents
- Directory Agents

    Functionality of SLP Directory Agents is not provided for Linux* or Solaris* systems.

User Agents

User Agents work in behalf of client applications to retrieve service URLs and attributes of desired network services. Client applications can request all URLs of a specific service type or narrow the search by requesting only services of a certain type with specific attributes.

If no Directory Agents are available to the User Agent, the SLP request is sent to multiple services (multicast) using the Service Location General Multicast Address (224.0.1.22, see RFC 2165). All Service Agents holding service information that satisfy the request unicast the reply (using UDP or TCP) directly to the requesting User Agent.

If a Service Agent has the requested service information, it replies. If multiple Service Agents reply, the User Agent combines the replies before presenting them to the client application. If a Directory Agent is available, the User Agent unicasts the SLP request to the Directory Agent rather than sending a multicast request. The Directory Agent always unicasts a reply even if the answer indicates that no services are available.

User Agents send the following SLP requests:

Table 115   SLP Requests Sent by User Agents

Request	Description
Service Type Request	Returns all active service types.
Service Request	Returns the service URLs of a specific type.
Attribute Request	Returns the attributes of a specific service URL.

User Agents process the following SLP replies:

Table 116   SLP Replies Processes by User Agents

Reply	Description
Service Type Reply	Contains the list of known service types.
Service Reply	Contains a list of the requested service URLs.
Attribute Reply	Contains the requested attributes of a specific service URL.
DA Advert	Sent by Directory Agents to indicate their existence.

Novell provides implementations of User Agents for NetWare, Windows 95/98, Windows NT, and Windows 2000.

Service Agents

Service Agents (defined by RFC 2609) work in behalf of network service applications to passively advertise service URLs representing the services provided. Network service applications register the service URL and attributes that define their network service with the Service Agent.

The Service Agent maintains a local database of registered service information. The Service Agent does not broadcast or multicast the registered services on the network but passively waits for SLP requests to be multicast from User Agents.

If Directory Agents are present, the Service Agent registers the services with each Directory Agent.

Service Agents send the following SLP requests:

- **Service Registration:** Registers a service URL and its attributes with a Directory Agent.

- **Service Deregistration:** Deregisters a service URL and its attributes from a Directory Agent.

Service Agents process the following SLP requests:

- **Service Type Request:** Returns all held service types.

- **Service Request:** Returns the service URLs of a specific type.

- **Attribute Request:** Returns the attributes of a specific service URL.

- **DA Advert:** Sent by Directory Agents to indicate their existence.

Novell provides implementations of Service Agents for NetWare, Windows 95/98, Windows NT, and Windows 2000.

Directory Agents

The Directory Agent maintains a database of service URLs representing network services. Service Agents acting in behalf of network applications register service URLs with the Directory Agent.

Multiple Directory Agents can be deployed in a network. Service Agents register their service URLs with each known Directory Agent, maintaining consistent service information among all Directory Agents.

RFC 2165 does not define a protocol for synchronizing service information between Directory Agents. To compensate, Novell SLP Directory Agents support a feature known as Directory mode.

Directory Agents configured for Directory mode use Novell eDirectory as a common, distributed, replicated data store through which multiple Directory Agents can share service URLs. This enables Directory Agents to report service URLs that were registered with other Directory Agents, configured in Directory mode, as well as report the services registered by local Service Agents.

Such reporting reduces network traffic by eliminating the need for Service Agents to register with every Directory Agent in the network. This reduction is particularly advantageous for large enterprise networks with WAN backbones.

Novell provides implementations of Directory Agents for NetWare, Windows NT, and Windows 2000. Directory Agents running on NetWare operate only in Directory mode. Directory Agents running on Windows NT or Windows 2000 can operate in Directory mode or Local mode. A Directory Agent operating in Local mode does not share service information with other Directory Agents. It operates autonomously, as defined by RFC 2165.

The Directory Agent is responsible for processing the following SLP protocol messages:

- Service Registration
- Service Deregistration
- Service Type Request
- Service Request
- Attribute Request
- Directory Agent Advertisements

These SLP messages enter, delete, or query for service URLs and associated attributes in the Directory Agent's service database.

For more information on these message types, refer to RFC 2165.

Service Registration

To register service URLs and their attributes with Directory Agents, Service Agents send Service Registrations. Each service URL includes a lifetime which, if it expires, causes the Directory Agent to delete the service from its database.

The Service Agent must refresh the service registration at least once during the service's lifetime. The service lifetime ensures that the Directory Agent can eventually purge its service cache of service URLs registered by Service Agents that do not deregister their service URLs.

Service Deregistration

To remove a service URL and its attributes from the Directory Agent service cache, Service Agents send Service Deregistrations to Directory Agents. This action can occur if the network application is terminating or if the Service Agent is being shut down.

Service Type Request

To obtain a list of active service types on the network, User Agents send Service Type Requests to Service Agents (multicast) or Directory Agents (unicast). Service Agents and Directory Agents return their known service types with a Service Type Reply, which is unicast to the requesting User Agent.

Service Request

Service Requests are sent by User Agents to Service Agents (multicast) or Directory Agents (unicast) in search of service URLs representing desired services. Service URLs matching the request criteria are returned in a Service Reply, which is unicast to the requesting User Agent.

Service Requests can be general in nature and request all URLs of a specific service type, or they can contain a predicate that specifies that only services of a certain type with specific attributes be returned.

Attribute Request

To retrieve one or more attributes of a specific service URL, User Agents send Attribute Requests to Service Agents (multicast) or Directory Agents (unicast).

The Attribute Request can be general, requesting that all attributes be returned. Also, the Attribute request can contain an Attribute Select list, identifying one or more specific attributes to be returned.

The requested attributes are returned in an Attribute Reply that is unicast to the requesting User Agent.

Directory Agent Advertisement

To periodically notify Service Agents and User Agents of Directory Agents' existence, Directory Agents multicast Directory Agent Advertisements. Directory Agents also return Directory Agent Advertisements in response to Service Requests for the directory-agent service type.

Directory Agent Advertisements contain:

 • The service URL for the Directory Agent

 • Other configuration information that helps User Agents and Service Agents determine which Directory Agents to direct SLP requests to

If multicasts are not enabled or allowed in a network, User Agents and Service Agents can be configured with the network addresses of Directory Agents. In such a case, the User Agent and Service Agent query (with a Service Request of type *directory-agent*) the Directory Agent for its Directory Agent Advertisement.

For a complete description of User Agent, Service Agent, and Directory Agent synchronization, see RFC 2165.

SLP Scopes

An SLP scope is a defined group of network services. Scopes enable one or more groups of users to easily use network services.

To define a scope, you can use criteria that help you organize and administer network services. If you have configured users to use a specific set of scopes, you can effectively assign a set of available services to those users.

You can create scopes to reflect departments in your company, for example:

 • A Human Resources scope groups the services unique to the Human Resources department

 • An Accounting scope groups the resources pertaining to the Accounting department

With these scopes, you can configure users in the Human Resources department to use the Human Resources scope. Also, you can configure users in the Accounting department to use the Accounting scope. Users requiring services in both departments can be configured to use both scopes.

Likewise, services can be grouped according to geographical location. You can define an SLP scope for each city or country where your company has an office. You can configure users in each locality to use the scope defined for their office. If a user needs access to services in multiple sites, you can configure that user to use the scopes of all necessary sites.

In addition to dividing services according to organizational and geographical criteria, you can define scopes to hold common services that multiple groups must share. This feature allows users to locate shared services while keeping their unique services locally.

Another reason to use scopes is to enhance the scalability and performance of SLP. Service registrations are organized and stored according to the scope in which they have been registered. Directory Agents are configured to service one or more scopes. If all services in a network are contained in a single scope, and hence a single service cache, the amount of service information can become unwieldy and difficult to manage. Response times might suffer because of the immense amount of data that must be searched to satisfy a request.

Therefore, in large network environments, it is better to group the services into scopes and then assign one or more Directory Agents to service the scopes applicable to the users that will be utilizing the Directory Agent.

Service Location Protocol 1 (RFC 2165) defines the default operating configuration of User Agents, Service Agents, and Directory Agents to be unscoped, meaning that no scopes are configured. This means that all services are maintained as if in a single scope that has no name.

 Additionally, special rules apply when registering with, or requesting services from, unscoped agents. In particular, all services regardless of scope should be registered with unscoped Directory Agents. But if an unscoped request is made to an unscoped agent, only those services registered as unscoped can be returned. On the other hand, a scoped request will return all services from the requested scope as well as all unscoped services matching the request criteria.

When both scoped and unscoped agents are used in the same network, results are often confusing and sometimes inconsistent. Therefore, Service Location Protocol 2 (RFC 2608) removed unscoped operation from the Service Location Protocol and redefined the default operating configuration to use a default scope named Default.

To eliminate the confusion associated with mixing unscoped and scope agents in a single network and to facilitate eventual migration to SLP 2, we recommend that users always configure SLP to use scopes.

For the following reasons, generally use scopes to organize SLP service:

- Services are registered into and retrieved from a scope.
- Many SLP configuration parameters are set according to scopes.
- Directory Agents are configured to service one or more scopes.
- User Agents and Service Agents determine which Directory Agent to query, based on the scopes the Directory Agent is supporting.

Fundamental to the successful organization, deployment, and administration of SLP in a network, scopes are a valuable tool in controlling the availability of services in the network

How SLP Works

User Agents and Service Agents interact on behalf of client applications and network services to dynamically locate network services.

Example with a User Agent, Service Agent, and No Directory Agent

Figure 36 illustrates how Service Agents and User Agents interact without a Directory Agent in the network. When a network application is started, it registers its service URL and attributes with the Service Agent. The Service Agent stores a copy of the service information in its local service cache. The Service Agent remains silent, meaning that the service is not multicast or broadcast on the network.

Figure 36   SLP User Agent and Service Agent Interaction.

When a client application queries the User Agent for a network service, the User Agent in search of service information multicasts a Service Request. The Service Agent receives the Service Request and consults its local service cache to see if it holds a service matching the criteria of the Service Request. If so, the Service Agent containing the requested service information unicasts a Service Reply to the User Agent.

If multiple Service Agents reply, the User Agent combines the results before returning them to the client application. This same scenario occurs for Service Type and Attribute Requests. When the network service is terminated, it deregisters its service with the Service Agent, which deletes the service from its local service cache. The Service Agent remains silent.

Example with a User Agent, Service Agent, and Directory Agent

Figure 37 illustrates how Service Agents and User Agents interact with Directory Agents to advertise and locate network services. When a network application is started, it registers its service URL and attributes with the Service Agent. The Service Agent maintains its own copy of the service information and then unicasts a Service Registration (that has the new service information) to the Directory Agent. The Directory Agent saves the service information in its local service cache.

Figure 37   SLP Directory Agent interaction.
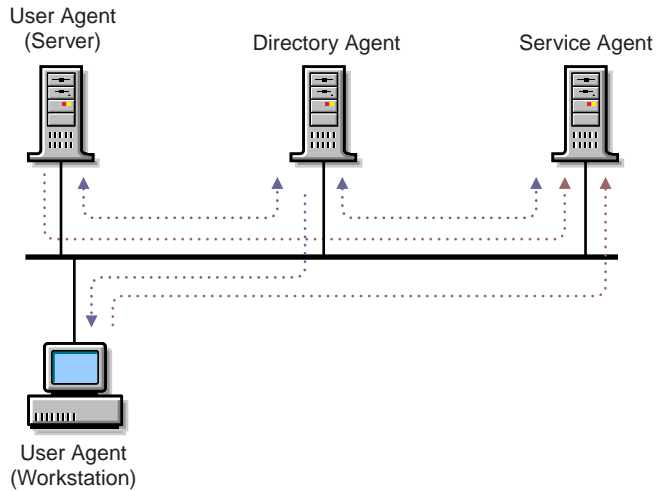
When a client application queries the User Agent for a network service, the User Agent in search of service information unicasts a Service Request to the Directory Agent. The Directory Agent returns a Service Reply that has the requested service URLs or an indication that no requested services are available. The same scenario is repeated by the User Agent and Directory Agent for Service Type Requests and Attributes Requests.

When the network service is terminated, it deregisters its service with the Service Agent, which deletes the service from its local service cache and then sends a Service Deregister request to the Directory Agent. The Directory Agent then deletes the indicated service from its service cache.

Understanding Local Mode

Novell Directory Agents can be installed and configured so that the Local mode operation can do the following:

- Provide a centralized repository of service URLs

- Facilitate the use of SLP scopes

- Create customized scopes by selectively gathering services from other scopes

- Proxy scopes directly supported by other Directory Agents or Service Agents

- Improve SLP scalability, performance, and network efficiency

- Facilitate the use of SLP in networks not supporting IP multicast

- Act as private Directory Agents for closed groups of Service Agents and User Agents through Private Mode

- Filter service content of SLP scopes based on service type, service URL, service lifetime, and the IP address of the Service Agent or User Agent

Central Repository

Directory Agents function as a centralized data store for service URLs that are registered by Service Agents and solicited by User Agents. Because Directory Agents hold all the services for each configured scope, User Agents can obtain all desired service information with a single request and reply. By contrast, in networks without Directory Agents, User Agents issue a multicast request and might receive many replies.

SLP Scopes

Directory Agents are configured to support one or more SLP scopes. (An unscoped operation is similar to supporting a single scope.) Directory Agents collect and store service URLs and their associated attributes according to the scope in which the services are registered. Service Agents and User Agents obtain the scopes supported by a Directory Agent from a Directory Agent's DA Advert message. In this way, User Agents and Service Agents can dynamically detect and utilize the scopes configured for each Directory Agent. In networks without Directory Agents, Service Agents and User Agents must be configured with the SLP scopes they will use.

Customized Scopes

Novell Directory Agents allow the network administrator to create customized scopes by pulling service information from one scope and storing it in a different scope. This is a variation of the scope proxy feature because the custom scope name is different than the scope being proxied.

For example, if a network administrator wants to create a custom scope for a single group of users containing only specific service URLs and attributes, the custom scope is configured on the local Directory Agent and the address of the scope authority servicing a target scope and the target scope's name is configured as a proxy address for the custom scope. The content of the custom scope can be further controlled by adding filters that apply only to the custom scope.

When the services are retrieved from the scope authority and registered in the custom scope, the attributes of the service are modified to indicate that the service is now part of the custom scope. The group of users can then be configured to use only the custom scope with the network administrator controlling the service information available to them. Using this same technique, a hierarchy of scopes can be created to reflect the administrative groupings of services that best fit your network user's needs.

Proxy Scopes

Novell Directory Agents can be configured to proxy scopes supported natively by other Directory Agents, also referred to as scope authorities. Instead of having every Service Agent register with every Directory Agent in the network, Service Agents can be configured to register with a single or small subset of Directory Agents. The other Directory Agents in the network are then configured to proxy the scopes of the central Directory Agents, which act as the authorities for the proxied scopes.

When a Directory Agent is configured to proxy a scope supported by another Directory Agent, the proxy agent downloads the scope information at configured intervals and then acts as a local service cache for that scope. This can be advantageous for remote sites reachable over WAN segments. Rather than having User Agents in remote sites interacting with Directory Agents over the WAN, a proxy Directory Agent can be deployed in the remote site, keeping all SLP service queries within the local site's network.

Scalability and Performance

Because service information can be registered and obtained with a single unicast request and reply, the operation of SLP becomes more efficient and hence more scalable. Because each interaction with a Directory Agent always results in a reply, the time required to resolve a service request is kept to a minimum. When a User Agent issues a multicast request, it must wait a period of time before determining if all answers have been received. This is because Service Agents and Directory Agents do not respond unless they can answer the query. As a result, the User Agent must pause while waiting for replies, estimating when all possible answers have been received. But as soon as a User Agent receives a reply from a Directory Agent it can process the response immediately.

All protocol interactions with a Directory Agent are performed using unicast messages. If multicast is not supported on your network, deploying a Directory Agent and configuring the Service Agents and User Agents with the IP address of the Directory Agent (through local configuration or DHCP) allows SLP to be used in networks that do not support multicast addressing.

Private Mode

In addition to the features listed above that are defined by the SLP protocol, Novell Directory Agents support other value-added features that assist the network administrator in deploying SLP within their network. Novell Directory Agents can be configured to operate in Private mode. When configured for Private mode, the Directory Agent does not multicast Directory Agent Advert messages or answer multicast requests, thus making the Directory Agent undiscoverable by dynamic means. To use a Directory Agent configured in Private mode, User Agents and Service Agents must be configured with the address of the private Directory Agent.

This allows the network administrator to create closed groups of users of one or more private Directory Agents. Private Directory Agents are also a valuable tool in piloting new versions of the Directory Agent or testing new configurations without disturbing the operating network.

Filtering

When a Directory Agent is operating in Local mode, network administrators can configure filters that control which service URLs are accepted for registration and which service URLs are returned in service replies. The filters are configured on a per scope basis, allowing network administrators to customize the content of each scope separately. The filtering criteria include service type, specific URLs, service lifetime, and the address of the Service Agent or User Agent making a request. One or more filter criteria can be specified for each filter.

Understanding Directory Mode

Novell Directory Agents can be configured for operation with eDirectory to:

- Provide a single point of configuration and administration of SLP agents

- Share service information among multiple Directory Agents

- Conserve network bandwidth

- Perform all operations supported by Local mode

SLP Directory Agents, scopes, and services can be configured and managed through eDirectory. This provides a single point of control to network administrators implementing and managing SLP in their networks.

Directory Agents are configured using Directory Agent objects containing configuration information for the Directory Agent. SLP Scope container objects can be configured to represent SLP scopes. A Directory Agent object contains fully distinguished names of one or more SLP Scope container objects which indicate the scopes the Directory Agent is to service. Services registered with the Directory Agent are stored in the SLP Scope container object as SLP Service objects. Each SLP Service object includes the service's service URL and attributes. SLP Service objects can be manipulated just like any other eDirectory object, including deleting and copying to another SLP Scope container object.

Novell Directory Agents can share service information by using eDirectory as a common data store for service URLs and their attributes. In this manner, the distributed, replicated, and synchronized nature of information stored in eDirectory is leveraged to eliminate the need for every Service Agent in the network to directly communicate with every Directory Agent in the network. SLP Scope container objects representing SLP scopes are configured in eDirectory. Directory Agents, configured to service the scope, cache each registered service locally and store each service and its attributes as an SLP Service object in the SLP Scope container object. These Directory Agents also populate their local service cache with services obtained from the SLP Scope container object. By storing and retrieving from the shared SLP Scope container objects, Directory Agents can return service URLs and attributes for services registered by remote Service Agents.

Because Directory mode facilitates the sharing of service information through common SLP Scope container objects, Service Agents are not required to register a service with every Directory Agent in order for their services to be known throughout the network. This reduces configuration complexity and reduces network traffic. Using this capability, Service Agent to Directory Agent interactions can be isolated to local segments within the network similar to User Agent to Directory Agent interactions.

How SLP Works in Directory Mode

A Novell Client™ uses the User Agent to go to an SLP Directory Agent or into eDirectory to reach out to other LAN or WAN segments, as shown in Figure 38 on page 388.

This method does not rely on service information obtained from routers. Instead, eDirectory is used for global communication of information. Through this method, service updates on local segments are just as reliable and dynamic as on IPX™ SAP-based networks.

Figure 38   Integrated Network Services Discovery

SLP eDirectory Objects

Using ConsoleOne™ you can manage the following eDirectory objects used by SLP:

- ◆ "SLP Scope Container Object" on page 389
- ◆ "SLP Service Object" on page 389
- ◆ "Directory Agent Object" on page 389
- ◆ "Server Object" on page 390

The SLP Scope container object represents an SLP scope and is the container in which SLP Service objects are stored.

SLP Service objects represent a network service discovered through the Service Location Protocol. They contain all of the SLP information about the network service, including its network address and attributes.

The SLP Directory Agent object represents an SLP Directory agent.

SLP Scope Container Object

SLP uses the SLP Scope container object, which defines a logical grouping of services. The Scope object allows network administrators to logically group services according to geographical, geopolitical, service type, or any other administrative criteria in order to control distribution or visibility on the network. The primary goal of the Scope object is to enhance the scalability of gathering and distributing network service information.

The SLP Scope object is the storage container for SLP service information. Each object contains all the SLP Service objects for the specific scope. The eDirectory administrator can replicate the container into other partitions within the tree or within federated trees. The object is a standalone entity within the eDirectory tree, and there is no relationship between its distinguished name, the tree name, and the scope name. When a Service Agent forwards a service record to a Directory Agent within a specific scope, the scope name is mapped to the Scope object by using the name attribute within the container object. The SLP Scope object must contain rights to read, write, and browse the container because the access rights of the Directory Agent object access are equivalent to the access rights of the Scope object. Because the Scope object uses distinguished name syntax, the Scope object can be moved to a different location in the tree. eDirectory will automatically change all values to reflect the new location.

SLP Service Object

The SLP Service object is a leaf object that represents a service registration. SLP Service objects are subordinate to the SLP Scope object and contain all information supplied by a service registration. SLP Service objects are stored in the appropriate SLP Scope object according to their scope.

Directory Agent Object

The SLP Directory Agent object is a leaf object that represents a single instance of a Directory Agent. Multiple Directory Agents cannot share a single object. This object defines the Directory Agent's configuration, scope, and security. The Directory Agent uses this object to log in to the server and operate under the access control requirements assigned to the Server object.

Server Object

The NetWare installation program creates an NCP_SERVER object for every server within the tree. The Directory agent adds an attribute to the NCP_SERVER class definition called SLP Directory Agent DN. The SLP Directory Agent DN contains the distinguished name of the Directory Agent object. It is used as a pointer from the Server object to the Directory Agent object.

Novell's Implementation of SLP

The following sections discuss Novell's implementation of the Service Location Protocol specification.

Novell's User Agents and Service Agents

The Novell Client includes software for User Agents and Service Agents. The software is installed automatically during a client installation when one of the IP protocol options is chosen.

SLP must be available for the client to function and should preferred for use before other Service Name resolving methods (that is, eDirectory, SAP, and so on) by the client. Otherwise, changing most of the SLP configuration parameters will have no functional effect on a Workstation/UA since it is either not available or is not being used to resolve service names.

To configure the parameters, go to the Novell Client Configuration property pages (right-click Network Neighborhood or My Network Places > click Properties > click Services > click Novell Client for Windows NT > click Properties).

SLP Configuration Parameters

Service Location Tab

The following paragraphs describe the options found on the Service Location tab of the Novell Client for Windows NT.

Scope List: Defines what SLP scopes the UA will participate in. Controls what DAs and SAs the UA will communicate with for SLP Service queries.

If the SA/DA is not in a scope specified at the UA, the UA will not send a request or accept a response from it. The exception to this is if there is no scope specified, then the UA will participate in the unscoped scope.

Scope entries can be set in a precedence order by using the up- and down-arrows. Scopes can come from three different sources: Static, DHCP, and Dynamic. As with other SLP settings, Static scopes have a higher preference than DHCP scopes, and DHCP scopes have a higher preference than Dynamic scopes for locating services.

Table 117 **Scope List Values**

Default Value	List is empty.
Valid Values	Any entry will be accepted, but should match the scope name used with the SAs/DAs you want to communicate with.

Static: Checking the Static check box will prevent the client from dynamically adding scopes discovered from the known active DAs. The active DAs can be checked by executing the SLPINFO command. If the Static check box is not checked, then when the client discovers a DA that participates in a scope that was previously unknown to the client, the client will add the scope to its list in memory and can now query for SLP Services in that scope.

Table 118 **Static Values**

Default Value	Unmarked (Off)
Valid Values	Marked/Unmarked (On/Off)

Directory Agent List: This parameter controls what DAs a client is statically configured to communicate with. This is not necessarily a complete list of the DAs the client is aware of. You must use SLPINFO with the /D command to be sure of what DAs the client has discovered and their status (Active/Inactive).

Table 119   **Directory Agent List Values**

Default Value	Empty
Valid Values	Any IP address or DNS-resolvable host name for a NetWare server running SLPDA.NLM

Static: Checking this check box will prevent dynamic DA discovery and only use DAs discovered using the Static or DHCP method.

- The UA will not send an initial DA multicast requesting a response from all DAs that can hear the request.

- Any DA that multicasts a DA Advertisement (DA_ADVERT) will be ignored. Normally the UA will add any DA that does a DA Advertisement. DAs Advertise when they are first loaded and also periodically based on the Heartbeat parameter.

Table 120   **Static Values**

Default Value	Unmarked
Valid Values	Marked/Unmarked (On/Off)

Active Discovery: Unchecking this check box requires that the UA contact a DA for an SLP Request. The UA will not multicast the request to SAs. The combination of Static enabled and Active Discovery disabled will entirely prevent the UA from multicasting. Once this setting is disabled, you are required to put at least one entry in the Directory Agent List; otherwise, the UA has no method for querying for SLP services.

Table 121   **Active Discovery Values**

Default Value	Marked (On)
Valid Values	Marked/Unmarked (On/Off)

Advanced Settings Tab

The following paragraphs describe the options found on the Service Location tab of the Novell Client for Windows NT.

Give Up on Requests to SAs: Timeout in seconds for an SLP Request to an SA. This parameter is not used to time out requests to DAs. There is a separate setting for this.

Table 122   **Give Up on Requests to SAs Values**

Default Value	15
Valid Values	1 - 60,000 seconds (16.67 hours)

SLP Cache Replies: Every time the UA receives an SLP Service reply from a DA/SA, it will be cached/saved at the UA for the amount of time specified in the SLP Cache Replies parameter. When SLP receives a request it will first check its cache before generating a network packet to a DA/SA. If the cached information can be used to answer the request, it will be. It is not recommended to set the time higher than one minute under normal SLP operations for the following reasons:

- During normal SLP communication, duplicate requests should occur within one minute of the original request, making longer caching unnecessary.

- The higher the setting, the more memory will potentially be required to cache this information.

Table 123   **SLP Cache Replies Values**

Default Value	1 minute
Valid Values	1 - 60 minutes

SLP Default Registration Lifetime: This parameter determines the registration lifetime of an SLP Service when an SA registers an SLP Service to a DA. The Novell Client not only includes the UA capabilities, but also the SA capabilities (the same as a server), so it is possible for a client workstation to be registering SLP services with a DA. However, it is unusual for a client workstation to be registering an SLP Service as an SA. Developers can write applications that register SLP Services from a client workstation using the WINSOCK 2 interface. Examples of cases where a client workstation would register an SLP service include:

- An NT Domain Controller running NDS4NT and a local eDirectory replica.

- A Client workstation running the Compatibility Mode client (CMD) where the workstation is advertising a SAP (for example, 0x0640). CMD will convert the SAP to SLP and register it with any DAs the client has discovered.

- Where a third-party application is using WINSOCK to intentionally register an SLP Service.

When the Registration Lifetime of an SLP Service expires, the DAs it is registered with will remove this entry from its database. This is also used to determine when an SA (on a workstation or a server) needs to re-register a service with its DAs.

Table 124   **SLP Default Registration Lifetime Values**

Default Value	10,800 seconds
Valid Values	60 - 60,000 seconds

SLP Maximum Transmission Unit: Exactly the same as the TCP/IP MTU, which is the maximum size that an SLP packet can have. This setting is used to restrict the size of the SLP packets so that it does not exceed the capability of the infrastructure and does prevent resource-intensive packet fragmenting and reassembly.

Table 125   **SLP Maximum Transmission Unit Values**

Default Value	1,400 bytes
Valid Values	576 - 4,096 bytes

SLP Multicast Radius: This parameter specifies the maximum number of subnets (number of routers plus 1) that SLPs multicasts can travel across. A value of 1 prevents multicasting from crossing any router. This is implemented in the Time To Live (TTL) setting of the UDP/TCP packet.

TTL is decremented by one of two conditions:

- The packet crosses a router.

- The packet is buffered in a router for more than 1 second.

Table 126   SLP Multicast Radius Values

Default Value	32 hops
Valid Values	1 - 32 hops

Use Broadcast for SLP Multicast: This parameter forces the SLP UA to use broadcast (all bits turned on in the Host ID portion of the address) where it would have normally used multicast.

This has the following different behaviors from multicast:

◆ Broadcast will not cross a router, so this limits the packet to the originating subnet.

◆ It might cause additional bandwidth usage because the packet will now need to be repeated from every switch port. Some switches are capable of tracking multicast registrations and would only forward a multicast packet from the switch ports that are registered for that multicast address.

Table 127   Use Broadcast for SLP Multicast Values

Default Value	Off
Valid Values	On/Off

Use DHCP for SLP: This parameter determines whether the SLP UA will attempt to locate a DHCP server that can provide SLP Scope and DA configuration information. Even if the workstation's IP address is statically configured, SLP can still receive an SLP Scope and DA configuration from a DHCP server. The DHCP requests for SLP information are sent only as part of the SLP UA/SA initialization. SLP information is requested using the DHCP INFORM request and is sent in addition to the initial BOOTP Request (if the client is configured to obtain its IP address via DHCP/BOOTP). All SLP DHCP response information is combined, then SLP contacts each DA that has been configured by DHCP to determine the scopes supported by each DA.

Administrators who plan to never use DHCP to administer SLP information should set this parameter to Off to reduce the minimal traffic the broadcast for a DHCP server will require.

Table 128   **Use DHCP for SLP Values**

Default Value	On
Valid Values	On/Off

Wait Before Giving Up on DA: Timeout in seconds for an SLP Request to a DA. This parameter is not used to time out requests to SAs. There is a separate setting for this.

Table 129   **Wait Before Giving Up on DA Values**

Default Value	5
Valid Values	1 - 60,000

Wait Before Registering on Passive DA: If an SA running on the workstation receives an unsolicited DA Advertisement (that is, either the DA just started or the DA issued a heartbeat), the SA will need to register whatever services it offers. This parameter is used to specify a range that the SAs will attempt to register their services to prevent the SAs on a network from all attempting to register with the DA at the same time. As mentioned earlier, the client workstation may need to use SLP to advertise Services it provides. This is unusual, but it may change in the future as applications begin to take advantage of this new advertising method.

Table 130   **Wait Before Registering on Passive DA Values**

Default Value	2 seconds
Valid Values	1 - 60,000 seconds

The Novell Directory Agent

The Service Location Protocol (SLP) Directory Agents support SLP 1. Enhanced features let network administrators better control the collection and dissemination of network service information through SLP.

Table 131   **Directory Agent Features**

Feature	Description	NetWare	Windows NT/ 2000
Directory-enabled operation	Directory mode uses eDirectory to store SLP service information.This leverages existing eDirectory standards for configuring eDirectory tree structures, for a central point of administration, and for the ability of eDirectory to replicate service information. eDirectory replication services allow Directory Agent-to-Directory Agent communication. This is unique in SLP implementations and it facilitates global distribution of SLP database information. eDirectory replica services give the Directory Agent the ability to access global services from a local replica. In Directory mode, you use ConsoleOne.	X	X
Local mode	Standalone operation. The SLP Directory Agent operates without using eDirectory. This lets network administrators use SLP Directory Agents in network segments that need the performance but don't need to share the service information globally (Windows NT Directory Agent Only). Use the SLP Directory Agent property pages on the Windows NT or Windows 2000 computer. For more information, see "Managing Properties for Local Mode" on page 407.		X

Feature	Description	NetWare	Windows NT/ 2000
Private mode	When operating in Private mode, the SLP Directory Agent only accepts SLP service registrations and requests from SLP agents configured with the SLP Directory Agent's IP address. In Private mode, the SLP Directory Agent does not multicast its presence on the network and does not answer multicast requests. For more information, see "Setting Up Private Mode" on page 410.		X
Proxy scope support	The SLP Directory Agent acts as proxy for scopes hosted by other SLP Directory Agents. This lets network administrators distribute service information from other SLP scopes, usually not visible to a local network segment, without having to enable network directory support. For more information, see "Setting Up a Proxy Scope" on page 408.		X
Service filtering support	The SLP Directory Agent can be configured with service filters that control the service information to and from SLP agents in the network. Additional filters can control the SLP service information that is stored in the network directory for global distribution. These filters provide single-point administration of the services made available through the SLP (Windows NT/ 2000 Directory Agent only). For more information, see "Setting Up Scope Filters" on page 409.	X	X

Using the Novell Windows NT Directory Agent

Scopes

In SLP, a scope is simply a list of SLP services that have been registered with a Directory Agent.

Using Scopes in Directory Mode

In Directory mode, when a Directory Agent is created, it registers the SLP Scope Unit container object, which is the actual storage container for SLP service information. Each Scope Unit container holds all the SLP Service objects for a specific scope. You can replicate this container into other partitions within the tree or within federated trees.

As mentioned earlier, the Scope Unit has an attribute called the Scope Name. This Scope Name is used by the Service Agent and User Agent to define what scopes they are to work with. SLP scopes allow network administrators to organize SLP services into groups. The Service Agent determines into which groupings the services on that the server will be registered. By default, all SLP services are registered in the unscoped scope. When clients send SLP requests to a Directory Agent, they can specify a scope for the Directory Agent to use in order to find the service they are looking for. If no scope is specified by the client, the Directory Agent will look in the Unscoped table to find the requested service.

A Directory Agent can service multiple scopes and a Service Agent can register services in multiple scopes. The registered services can be replicated between sites by using eDirectory.

Using Scopes in Local Mode

Scopes configured in Local mode operate similarly to scopes configured for Directory mode with the exception that the scopes are stored locally instead of in eDirectory. By default, all SLP services are registered in the unscoped scope. We recommend that you configure at least one scope.

For more information about configuring scopes in Local mode, see "Adding a New Scope" on page 408.

Using Scopes to Handle the 64 KB Limitation Issue

A total of 64 KB of data is all the Directory Agent can send to the client via a TCP connection. If there is more than 64KB of a certain service type, the list will be cut short. The reason for this is that in SLP 1, the length field in the SLP response packet header is only 16 bits, allowing up to 64 KB of service data.

Table 132 lists the common service types that fit into a 64 KB response packet.

Table 132   Common Service Types

Service	Number per Response Packet
NDAP.Novell	About 1,200, depending on the length of the partition names
Bindery.Novell	700 - 1,100, depending on the length of the server names
MGW.Novell	About 1,200
SapSrv.Novell	No more than 540

Understanding Scope Filtering

SLP uses scopes to logically group services according to administration, usage, or service type criteria. By dictating the scopes that SLP User Agents and Service Agents participate in, you can control the service information users see. Unfortunately, that level of control is not sufficient for large and sophisticated network environments. To give you better control over the collection and distribution of service information, use the additional filtering capabilities provided as part of the SLP Directory Agent configuration and management tools.

When administering scopes, you can configure registration, response, and directory filters for each scope.

- Registration filters restrict or control the service information that is accepted and stored by the Directory Agent for a given scope.

- Response filters restrict or control the service information that is returned to specific users or groups of users.

- Directory filters control whether the service information that is registered with the Directory Agent (subject to the registration filters) is also stored in the corresponding Scope Unit container object.

The Registration, Response, and Directory filters are configured on a per-scope basis. This lets you separately control the type of information stored in each scope.

Filtering

The SLP Directory Agent can be configured with service filters that control the service information to and from SLP agents in the network. Additional filters can control the SLP service information that is stored in the network directory for global distribution. These filters provide single-point administration of the services made available through SLP (Windows NT/2000 Directory Agent only).

Using INCLUDE and EXCLUDE Filters

Registration, response, and directory filters are specified using the INCLUDE and EXCLUDE filter directives.

The INCLUDE filter directive specifies criteria that the service registration or request must comply with to store or retrieve service information in the specified scope.

The EXCLUDE filter directive specifies criteria that prohibit any compliant service registration or request from occurring for the specified scope.

The filters associated with a scope consist of one or more INCLUDE and EXCLUDE filter directives. For a service registration or request to be processed, it must match at least one INCLUDE filter directive and not match any EXCLUDE filter directives configured for the scope. If any INCLUDE directives are configured, only service registrations and requests matching at least one INCLUDE directive are processed; all others are denied. If no INCLUDE directives are configured, all service registrations and requests are processed subject to any EXCLUDE filter directives.

The criteria for an INCLUDE or EXCLUDE filter directive are specified by one or more filter operations. Filter operations allow the administrator to filter the type of service, the specific service URLs, a service registration's lifetime, or the address of the sending or requesting host in the network. If you specify multiple filter operations in a single filter directive, all filter operations must evaluate to TRUE for the filter directive to be TRUE. Only one of each type of filter operation can be included in a single filter directive.

If the IP address of the sending or requesting host is used as filter criteria, it is specified in dotted decimal notation (for example, 137.65.143.195). Subnet masks can be associated with an IP address by appending a slash (/) followed by the subnet mask. The subnet mask can be specified using either dotted decimal notation or by specifying the number of contiguous bits that constitute the mask (for example, 137.65.143.0/255.255.252.0 and 137.65.143.0/22 are equivalent). If a subnet mask is specified, the mask will be applied to both the address specified in the ADDRESS filter operation and the host IP address being checked before any filter evaluations are made.

Filter Syntax

The ABNF (RFC 2234) for the registration, response, and directory filters is defined below:

```
Registration Filter = 1*(include_directive /
exclude_directive)
Response Filter = 1*(include_directive / exclude_directive)
Directory Filter = 1*(include_directive / exclude_directive)
include_directive ="INCLUDE("filter_operation")"
exclude_directive = "EXCLUDE("filter_operation")"
filter_operation = [address_operation] [type_operation]
[lifetime_operation]   [url_operation]
address_operation = "(ADDRESS" equality_operator *1(
ipv4_number / ipv4_number "/" subnet_mask )")"
lifetime_operation = "(LIFETIME" filter_operator seconds")"
type_operation = "(TYPE" equality_operator [wild]
service_type [wild]")"
url_operation = "(URL" equality_operator [wild] service_url
[wild]")"
service_url = service: URL as defined by RFC 2609
service_type = abstract-type ":" url_scheme / concrete-type
abstract_type = type_name ["." naming_auth]
concrete_type = protocol ["." naming_auth]
type_name = resname
naming_auth = resname
protocol = resname
url-scheme = resname
wild = "*"
reserved = "(" / ")" / "*" / "\"
escaped = "\" reserved
resname = ALPHA [1*(ALPHA / DIGIT / "+" / "-" )]
ipv4_number = 1*3DIGIT 3("." 1*3DIGIT)
subnet_mask = ipv4_number / 1-32
equality_operator = "==" | "!="
filter_operator = "==" / "!=" / ">" / "<"
seconds = 1-65535
```

Examples of INCLUDE and EXCLUDE Filter Directives

Below are examples of INCLUDE and EXCLUDE filter directives to help you understand how to implement the filter feature.

- Registration Filters

  Allow only services of types ndap.novell or bindery.novell with a lifetime greater than 5,000 seconds from servers on the 137.65.140.0 subnet to be stored by the SLP Directory Agent. The ADDRESS operation values for both INCLUDE directives are equivalent. The first registration filter uses dotted decimal notation for the subnet address and the second registration filter specifies the number of bits in the subnet mask.

  ```
  INCLUDE((TYPE == ndap.novell)(ADDRESS == 137.65.140.0/
  255.255.252.0))
  INCLUDE((TYPE == bindery.novell)(ADDRESS == 137.65.140.0/
  22))
  EXCLUDE ((LIFETIME < 5000))
  ```

- Response Filters

  Prevent only workstations on the 137.65.140.0 subnet (except the workstation with IP address 137.65.143.155) from accessing information held by the SLP Directory Agent.

  ```
  INCLUDE((ADDRESS == 137.65.140.0/255.255.252.0))
  EXCLUDE((ADDRESS == 137.65.143.155))
  ```

- Directory Filters

  The first two directory filters allow only services of types ndap.novell and bindery.novell to be stored in the Scope Unit container object associated with this scope. The second two directory filters allow only services with the URLs specified to be stored in the Scope Unit container object associated with this scope.

  ```
  INCLUDE((TYPE == ndap.novell))
  INCLUDE (TYPE == bindery.novell))
  ```

  or

  ```
  INCLUDE((URL == service:ndap.novell:///
  GLOBAL_PARTITION1.CORP_TREE.))
  INCLUDE (URL == service:ndap.novell:///
  GLOBAL_PARTITION2.CORP_TREE))
  ```

When the Directory Agent is operating in Local mode, the registration, response, and directory filters are stored in the local system's registry and are persistent across system boots.

When the Directory Agent is operating in Directory mode, the registration, response, and directory filters are stored as part of the Scope Unit directory object defining the filtered scope. The Scope Unit object has a Registration Filters, Response Filters, and Directory Filters attribute. These attributes are multi-valued of type SYN_CI_STRING. Each INCLUDE and EXCLUDE filter directive is stored as a separate string in the Registration Filters, Response Filters, or Directory Filters attribute.

Using the Service Location Protocol Directory Agent

The following scenarios show some of the many options for deploying SLP.

Scenario 1: Remote Site with a Mixed NetWare and Windows NT Environment

Problem: A remote office is running NT servers and NetWare clients with no NetWare servers. The administrator wants the clients to see all the network services from a local server, avoiding sending on-demand service queries over the slow link.

Solution: The Directory Agent can be installed on a Windows NT server to allow the clients to see all the network services from a local server without causing on-demand traffic over the slow link.

Scenario 2: Remote Office with Windows NT Servers Only

Problem: A remote office is running NT servers, and the administrator wants local clients to see only a limited set of services.

Solution: Use the new Directory Agent and its filter or proxy capabilities to configure the Directory Agent to only see a specific set of services.

Scenario 3: Using the Directory Agent for a Small Group of Users

Problem: An administrator wants to configure a Directory Agent for a group of users and wants that Directory Agent to manage only a small subset of services, not all SLP services on the network.

Solution: The administrator defines exactly which services are allowed to register with that Directory Agent. Then, by statically assigning the Directory Agent's address to those users, the administrator controls which services are seen by those users.

Scenario 4: Restricting SLP Information

Problem: An administrator wants to restrict the users who can query SLP information from a Directory Agent.

Solution: Set the filters on the Directory Agent for Windows NT to define who can obtain information from the Directory Agent. This identification is determined by the IP address.

Scenario 5: Synchronizing SLP Information Over a WAN Link

Problem: An administrator wants to synchronize SLP service information over a WAN link, but one side of the link uses eDirectory without any NetWare servers.

Solution: Run the Directory Agent on a Windows NT server and configure the Directory Agent to service the eDirectory scope containers included in the network eDirectory replication design.

Scenario 6: Replicating SLP Information to a Remote Site

Problem: An administrator wants to replicate SLP service data to a remote site without using eDirectory as the replication method.

Solution: The Directory Agent is installed on a Windows NT server at the remote site and is configured to proxy the data in another Directory Agent's scope. The Directory Agent scope that contains the original service information is known as the Scope Authority. The Directory Agent at the remote site is configured to look at a Scope Authority and can replicate the data to the remote site by using standard SLP requests to the Directory Agent.

Scenario 7: Running a Directory Agent in Local Mode

Problem: An administrator needs SLP on the network to find printers and other services. The administrator needs a Directory Agent to handle unicast requests since multicast packets are disabled on the network, and unicast is more efficient in bandwidth use.

Solution: Run the Directory Agent for Windows NT in a Local mode of operation. The services are only stored in memory and not in a Directory Service. This means that the Directory Agent can be run on Windows NT without the Novell Client or eDirectory.

Scenario 8: Using the Proxy Feature

Problem: An administrator of a development group notices that services are going up and down. The administrator wants a more active method of making sure the service information in SLP is accurate instead of relying on the default service lifetime protocol.

Solution: Use the proxy feature in the Directory Agent for Windows NT to configure the Directory Agent to poll another Directory Agent or Service Agents's scope. Configure the Directory Agent with Service Agent IP addresses as the Scope Authorities. This causes the Directory Agent to poll each Service Agent at a configured interval, querying for all active services.

Setting Up SLP on Windows NT or Windows 2000

This section explains how to set up SLP on a Windows NT or Windows 2000 system.

Installing the Windows NT/Windows 2000 Directory Agent

1 On a Windows NT or Windows 2000 machine, insert the *Novell eDirectory* CD.

2 Click Start > click Run > click Browse > select SETUP.EXE from the \NT directory of the CD.

3 From the installation screen, click SLP Directory Agent > Install.

Follow the online instructions for the SLP Directory Agent installation program.

If you select a Directory type of setup, the eDirectory schema will be extended for the eDirectory tree specified.

Managing Properties for Local Mode

1 On the server where the Directory Agent is running, click Start > Programs > Novell SLP Directory Agent > SLP DA Properties.

2 Adjust the configuration properties for the Directory Agent.

Adding a New Scope

To add, delete, or modify a scope:

1 On the server where the Directory Agent is running, click Start > Programs > Novell SLP Directory Agent > SLP DA Properties.

2 Click Scopes > Add.

or

Select an existing scope > click Properties to modify the scope or Delete to delete the scope from the list.

3 Enter the name of the new Scope > click OK.

For more information about setting up scope filtering, see "Setting Up a Proxy Scope" on page 408.

For more information about setting up a proxy scope, see "Setting Up Scope Filters" on page 409.

Setting Up a Proxy Scope

To add or delete a scope:

1 On the server where the Directory Agent is running, click Start > Programs > Novell SLP Directory Agent > SLP DA Properties.

2 Click Scopes > select the scope you want to add a proxy to from the list.

3 Click Properties > Proxy.

4 Enter the name of the Scope Authority to be proxied > click Add.

The syntax of the Scope Authority specification is as follows:

```
scope_authority [/[refresh_interval] [/
[character_encoding] [/target_scope]]]
```

The variables for this syntax include the following:

Table 133   **Scope Authority Variables**

Variable	Description
Scope_authority	The IP address or DNS name of the Directory Agent acting as the scope authority for the proxied scope.
Refresh_interval	The amount of time, in minutes, service information from this scope authority is to be retrieved. This value overrides any other refresh interval time configured for the scope but only applies to this scope authority.
Character_encoding	Indicates the character encoding to use when retrieving service information from this scope authority. Character encodings include ASCII, UTF8, and Unicode*.
Target_scope	The name of the scope to be queried for service information. If this value is omitted, the current scope name is used.

Setting Up Scope Filters

To add, delete, or modify scope filters:

1 On the server where the Directory Agent is running, click Start > Programs > Novell SLP Directory Agent > SLP DA Properties.

2 Click Scopes > select the scope you want to add the filter to from the list.

3 Click Properties > Filters.

4 Select the type of filter you want to add > click Add.

5 Select the filter parameters you want to include and exclude.

For more information about the filtering, see "Understanding Scope Filtering" on page 400.

Setting Up Private Mode

Private mode allows you to limit the visibility of the Directory Agent to those services configured specifically with the Directory Agent's IP address.

1 On the server where the Directory Agent is running, click Start > Programs > Novell SLP Directory Agent > SLP DA Properties.

2 Check the Private Mode check box to enable or disable Private mode.

Managing the Directory Agent in Directory Mode with ConsoleOne

To manage a directory agent running in Directory mode, use ConsoleOne. You can install ConsoleOne from the eDirectory CD.

- ◆ "Setting Configuration Properties in Directory Mode" on page 410
- ◆ "Adding a Serviced Scope Unit" on page 410
- ◆ "Setting Up Scope Filters" on page 410

Setting Configuration Properties in Directory Mode

1 In ConsoleOne, double-click the SLP Directory Agent object you want to modify.

The Properties page for the Directory Agent appears.

2 Adjust the settings.

Adding a Serviced Scope Unit

1 In ConsoleOne, double-click the SLP Directory Agent object you want to modify.

The Properties page for the Directory Agent appears.

2 Click the SLP Scope Units tab > Add.

3 Browse the eDirectory tree > select the Scope Unit you want to add.

Setting Up Scope Filters

1 In ConsoleOne, right-click the SLP Scope Unit object > select Properties.

The Properties page for the SLP Scope Unit appears.

2 Click the Filters tab > select the type of filter you want to create.

3 Click Add > enter the filter information you want.

For more information about filtering, see "Setting Up Scope Filters" on page 409.

Setting Up SLP on NetWare

This section explains how to set up SLP on a NetWare Server.

Installing the NetWare SLP Directory Agent

The software to implement SLP on NetWare is installed on the server during the server installation.

To set up SLP:

1 At the server console, type **LOAD SLPDA**.

   The program searches eDirectory for a SLP Directory Agent. If an agent does not exist, the program reports that a SLP Directory Agent has not been configured.

2 Press Enter to setup a default configuration.

   The schema is extended and a Directory Agent object with the name *server_name*_SLPDA and a Scope Unit named SLP_SCOPE are created and linked. This is recommended unless you want to create the SLP objects in eDirectory yourself.

   Use ConsoleOne to adjust settings for the Directory Agent object.

Setting Up the NetWare Directory Agent Manually

To configure SLP using ConsoleOne:

1 Start ConsoleOne.

2 Select the container where you want the SLPDA to reside.

3 Click Object > Create > SLP Directory Agent > OK.

4 Type the Directory Agent object name > click Define Additional Properties > click Create.

5 Select a host server.

6 Select the container where you want the Scope Units stored.

7 Click Object > Create > SLP Scope Unit > OK.

8 Type the name for the SLP Scope Unit.

9 Double-click the SLP Directory Agent object.

10 Click the SLP Scope Units page > Add.

11 Select the scope units serviced by this Directory Agent.

NetWare SLP Directory Agent Console Commands

Table 134 lists the SLP commands:

Table 134   SLP Commands

SLP OPEN *filename.log*	The SLP trace file is created in the root of the volume SYS:.
SLP CLOSE	This command closes the SLP trace file.
DISPLAY SLP SERVICES	Common Novell SLP service types include the following:
	MGW.NOVELL (Compatibility mode gateway/migration agents)
	CMD.NOVELL (Compatibility mode server/relay agents)
	NDAP.NOVELL (NDS)
	BINDERY.NOVELL (NetWare servers)
	SAPSRV.NOVELL (NetWare 5 or later servers with IPX CMD loaded)
	RMS.NOVELL (Resource Management Service of NDPS)
	RCONSOLE.NOVELL (Java* RCONSOLE)
	SRS.NOVELL (NDPS broker)
	DIRECTORY-AGENT (sends an SLP multicast packet to rediscover the DA on the network)
	SLP restrictions are as follows:
	slp_attribute==value
	Other operators available are <=, >=

	Examples of using the Display SLP Services command include the following:
	DISPLAY SLP SERVICES (Displays all known SLP services)
	DISPLAY SLP SERVICES BINDERY.NOVELL (Displays all bindery.novell services)
	DISPLAY SLP SERVICES BINDERY.NOVELL//(SVCNAME-WS==ABC*)/ (Displays bindery.novell services with names that begin with abc)
	DISPLAY SLP SERVICES BINDERY.NOVELL/PROVO/ (SVCNAME-WS==ABC*)/ (Displays bindery.novell services with names that begin with abc in scope provo)
	DISPLAY SLP SERVICES MBW.NOVELL//(CMD NETWORK==ABC12345)/ (Displays all the Migration Agents servicing the CMD network number ABC12345)
DISPLAY SLP ATTRIBUTES (*SLP_URL*)	The following is an example of using the Display SLP attributes command:
	DISPLAY SLP ATTRIBUTES SERVICE:BINDERY.NOVELL:/// SERVER1 (Displays all SLP attributes and values for the SERVER1 bindery.novell object)
DISPLAY SLP DA	Displays the list of SLP Directory Agents and their current status

NetWare SLP Directory Agent SET Commands

Table 135   SLP Directory Agent SET Commands

SET SLP DA Discovery Options = *value*	where *value* = 0 to 8 (Default = 3)
	0x01 = Use multicast DA advertisements
	0x02 = Use DHCP discovery
	0x04 = Use static file SYS:ETC\SLP.CFG
	0x08 = Scopes Required
SET SLP TCP = *value*	where *value* = ON/OFF (Default = OFF) This sets SLP to use TCP packets instead of UDP packets when possible.
SET SLP DEBUG = *value*	where *value* = 0 to 4294967255 (Default = 0)
	0x01 = COMM
	0x02 = TRAN
	0x04 = API
	0x08 = DA
	0x010 = ERR
	0x020 = SA
	These bits can be combined with AND or OR statements for multiple values. An example of COMM and API would be 0x05.
SET SLP Multicast Radius = *value*	where *value* = 0 to 32 (Default = 32)
	This parameter specifies an integer describing the multicast radius.

SET SLP Broadcast = *value*	where *value* = ON/OFF (Default = OFF) This parameter sets the use of broadcast packets instead of multicast packets.
SET SLP MTU size= *value*	where *value* = 0 to 4294967255 (Default = 1472) This parameter specifies an integer describing the maximum transfer unit size.
SET SLP Rediscover Inactive Directory Agents = *value*	where *value* = 0 to 4294967255 (Default = 60) This parameter specifies the minimum time period in seconds that SLP will wait to issue service requests to rediscover inactive directory agents.
SET SLP Retry Count = *value*	where *value* = 0 to 128 (Default = 3) This parameter specifies an integer value describing the maximum number of retries.
SET SLP Scope List = *value*	where the maximum length of *value* is 1023 (Default = 1023) This parameter specifies a comma-delimited scope policy list.
SET SLP SA Default Lifetime = *value*	where *value* = 0 to 4294967255 (Default = 900) This parameter specifies an integer value describing the default lifetime in seconds of service registers.
SET SLP Event Timeout = *value*	where *value* = 0 to 4294967255 (Default = 53) This parameter specifies an integer value describing the number of seconds to wait before timing out multicast packet requests.
SET SLP DA Heart Beat Time = *value*	where *value* = 0 to 4294967255 (Default = 10800) This parameter specifies an integer value describing the number of seconds before sending the next Directory Agent heartbeat packet.
SET SLP Close Idle TCP Connections Time = *value*	where *value* = 0 to 4294967255 (Default = 300) This parameter specifies an integer value describing the number of seconds before idle TCP connections are terminated.

SET SLP DA Event Timeout = *value*	where *value* = 0 to 429 (Default = 5)
	This parameter specifies an integer value describing the number of seconds to wait before timing out Directory Agent packet requests.
SET SLP Maximum WTD = *value*	where *value* = 1 to 64 (Default = 10)
	This parameter specifies the maximum number of work-to-do threads that SLP can allocate.
SET SLP Reset = *value*	where *value* = ON/OFF
	(Resets to OFF each time it is set to ON) This parameter forces the SA to send new service registers and forces the SA to send DA Advertise packets.
SET SLP Debug = *value*	where *value* = 0 to 65535 (Default = 88)
	0x01 = COMM
	0x02 = TRAN
	0x04 = API
	0x08 = SA_DA
	0x010 = ERR
	0x020 = SA
	0x040 = UA_DA
	These bits can be combined with AND or OR statements for multiple values. An example of COMM and API would be 0x05.

Setting Up SLP on Linux or Solaris

User Agents and Service Agents.

The slpuasa deamon provides the functionality for both the User Agent as well as the Service Agent.The slpuasa deamon will read the /etc/slpuasa.conf file on start-up and do a Active-Discovery of the DA on the Network if the "isMulticastOnly" option is disabled or commented (Default value is 0), otherwise the slpuasa deamon will use the multicast option.

The slpuasa.conf file contains a list of configuration parameters for configuring the SLP User Agent - Service Agent daemon. slpuasa daemon reads this file when it is started.

If any change is made to the slpuasa.conf file, the daemon needs to be restarted for the changes to be reflected.

Each parameter is in a single line in the form: parameter=value

Parameters are described as follows:

net.slp.MTU

The Maximum Path Unit to be set parameter is used by the Service Agent to service the SLP requests. The default value is 1400.

net.slp.isMulticastOnly

A value of 1 means that the slpuasa daemon will not discover any Directory Agents (active or passive). Use this option only if you have problems due to a Directory Agent, because this parameter can cause an unnecessary increase in the multicast traffic in the network. The default value is 0.

net.slp.isBroadcastOnly

Use to select either broadcast or multicast SLP requests. A value of 0 means the User Agent and Service Agent use multicast. A value of 1 means they use broadcast. The default value is 0.

net.slp.MulticastRadius

The value of the site's multicast TTL. The default value is 32.

net.slp.useScopes

SLP scope that the preferred DA is configured with. Provides the Scopes with which this UASA is configured with the preferred DA.

DA_ADDR

Preferred DA IP address or DNS name.

Example :

#net.slp.useScopes = printer.

About Starting and Stopping the Deamon :

The slpuasa can be started/stooped with the slpuasa script as /etc/init.d/slpuasa {start/stop} on Solaris and as /etc/rc.d/init.d/slpuasa {start/stop} on Linux.

Information About the SLPINFO Diagnostic Utility :

The slpinfo diagnostic Utility helps to list the :

a). -t Option : Lists the various servie-types.

b). -s <service-type> : Lists Various Services Under the Service-type :

example : slpinfo -s "ndap.novell///" will list all the services under the Service-Type : "ndap.novell".

The query can also be given with attributes :

Example : slpinfo -s "ndap.novell//svcname-ws==*name*.)" will list all the services containing text name.

c). -a "service:ndap.novell:///treename."

The query will list the Attributes of the respective-tree. i.e. svcname , svcaddr (The TCP and UDP addresses of the Server) , svcid , version , nds-version etc.

12 Backing Up and Restoring Novell eDirectory

The best way to protect your database is to use replicas. A tape backup that provides a snapshot in time also increases the fault tolerance for your network. Tape backups protect data and Novell® eDirectory™ information in single-server environments and in the event of a catastrophe such as a fire or flood. Replication, however, is not sufficient protection for a single-server network or when all copies of the replicas are destroyed or if one of the replicas become corrupted. In these instances, if the data has been backed up regularly, the tree structure can be restored using any backup/restore utility compliant with SMS™.

Novell provides the following SMS-compliant utilities to back up and restore on each platform:

   ◆ SBCON.NLM on NetWare® 6

   ◆ SMSENGN.EXE on Windows* NT*

   ◆ ndsbackup utility on Linux* and Solaris*

Understanding Backup and Restore Services

To archive or restore eDirectory objects, specify the fully distinguished name of a leaf object or a container to be archived, extracted, or listed. To archive the whole tree, specify the Tree object. You can also back up the schema by specifying Schema as the object.

Backup Services

You can back up the entire tree or a part of the tree starting with a particular container. You can back up the schema and schema extensions.

You cannot back up partition information. If the tree structure becomes corrupted and you restore the data, all data is restored to one partition, the Tree partition. You need to repartition that portion of the tree. You should keep a written copy of the tree structure and the partitions.

You can begin the backup of the database from anywhere in the tree structure. The backup process continues from that point downward to the end of that portion of the tree. If the selected container is Tree, the entire tree structure is processed. This allows you to back up the entire tree structure or subsets such as a single branch, a single container, or even a single leaf object.

When you back up eDirectory, we recommend that you back up the tree structure in one session. Partial eDirectory backups and restores are possible, but they are more difficult.

Customizing Your Backup

The backup utility lets you customize the backup process. You can choose specific eDirectory objects to exclude from or to include in the backup session. You might want to use include or exclude if your tree is spread across several geographic locations. In this scenario, a full directory backup crossing WAN links could result in a noticeable performance hit.

Whether you use Exclude or Include usually depends on the size of the data you want to back up, compared to the size you do not want to back up. By combining the Exclude and Include options, you can control what is backed up.

Exclude

To back up most of the tree structure while omitting only a small part, use the Exclude option to omit the part you do not want to back up. You can exclude objects by distinguished name or exclude a subtree by a container name. Everything that you do not want specifically excluded is included. After you exclude part of the structure, you cannot include objects below that container.

Include

To back up a small part of the tree structure, use the Include option to specify the data you want. You can include objects by distinguished name or include a subtree by a container name. Everything you do not want specifically included is excluded.

When you specifically Include a subtree by container name, all objects below that container are included.

Frequency of Backups

In general, the database should be backed up on a weekly basis. The frequency of this backup depends on how often changes and updates are made to the tree structure. For a tree that changes often, you might want to perform an eDirectory backup every time you do a full backup of all the servers on the network.

IMPORTANT: Always back up eDirectory prior to major tree modifications. To get a full backup, the entire tree structure needs to be functioning, meaning that all partitions are synchronized normally. A tree cannot be entirely backed up if any replicas of any partition are offline.

Restore Sessions

You can restore objects if they have been lost or corrupted since a backup was made. A restore session restores data from a backup. The restore session retrieves the requested objects from the backup file and restores them to the location you specify. For a custom restore session, you can specify exactly which data to restore. Several options work together for maximum flexibility in a restore session.

Restoring eDirectory

The best way to ensure that your database can be fully protected is through partition replication, with replicas of the entire database on multiple servers. However, on a single-server network, you must rely on backing up the data because you do not have the replicas to restore information.

 If part of the tree structure, including partitions and replicas, exists when the database information is restored, those partitions and replicas will left as they are, and you will not need to repartition the tree.

In case of corrupted data, follow these general steps:

1 Delete the corrupted data.

2 Allow time for the deletion to propagate throughout the network.

The allotted time depends on the size of the data to be backed up, the size of your network, the number of servers you have, and the number of containers and users you have.

3 Restore the data.

A replica containing the object does not have to be on the server. The database creates an external reference when necessary.

An external reference is a pointer to an object not found locally on the server; it is used to authenticate and reference objects that are not local to the server.

Subsets of Data to Restore

You can choose specific subsets of a backup session to include in or exclude from the restore session by selecting containers or objects. For more information about including and excluding, see "Customizing Your Backup" on page 420.

Partially Restoring eDirectory

The backup utility lets you perform selective restores from the backup file. However, partial eDirectory restoration from a backup can have many subtle consequences, particularly when only a single object or a selected group of these objects is restored.

For partial eDirectory restores, keep these two main issues in mind:

◆ **Object ID Numbers:** If you restore objects that no longer exist in the tree, those objects receive new ID numbers when they are restored. New object IDs affect file system trustees, print queue directories, user mail directories, etc.

If you restore objects on top of objects that exist in the tree, the objects do not receive new ID numbers. These objects' current attribute and property information is overwritten with previous information from the backup.

◆ **Objects That Depend on Other Objects:** In the schema, objects are defined to have certain attributes. Some of these attributes are mandatory (meaning they must contain a value); others are optional.

For some objects, the value for a particular attribute is a reference to another object upon which the object depends. For example, the Queue object has a Queue directory attribute that contains the file system to the queue directory. It also has a Host Server attribute that identifies the file server on which the queue directory resides. This information is used to determine the physical location of the resource.

The specifics of restoring objects vary depending on what type of object is involved and whether the object's dependencies are physical entities (servers and volumes) or logical entities. In some cases, an object might be restored but not be functional unless you first restore its dependent objects.

Overwriting Existing eDirectory Objects

Be careful when you perform a selective restore, overwriting existing eDirectory objects. Objects such as groups and users have references to other objects in the tree structure that will be affected by a selective restore.

For example, suppose a part of the tree structure gets corrupted and several users are deleted from the tree. There is a group that contains those users, but once the users are gone, the group purges the membership list to remove those users; the group, however, continues to exist in the tree structure.

If you perform a selective restore and choose not to overwrite existing objects, the group membership list remains empty even if you restore users. You need to either add the users manually to the group membership list or restore the original group.

Using Backup and Restore Services on NetWare

For instructions on backing up and restoring eDirectory on a specific server rather than the entire tree, refer to and .

Use any SMS-compliant backup/restore utility to manage tape devices, target services, and backup and restore options. The following components are essential for backing up and restoring eDirectory on a NetWare server.

- ◆ The Storage Management Engine (SME) is the back-end engine that processes backup/restore requests. SBCON.NLM is a basic SME provided by Novell for NetWare 6. If you are already using an SMS-compliant backup engine based on NetWare, you should continue to use it. If you use SBCON, then you should first load QMAN.NLM, which will manage the backup and restore jobs created in SBCON.

- The Storage Management Data Requester (SMDR) communicates between the SME and the Target Service Agent (TSA) software. The first time that SMDR.NLM is loaded on a server in a directory, it prompts you for several configuration options, including the option to set up an SMDR object in the directory tree.

  The SME and TSA can be on two different computers or on the same computer. If the SME and TSA are on different computers, then an SMDR is required on both machines. SMEs are available on NetWare, Windows NT, and Windows. SME on NetWare creates backups to a tape.

- Target Service Agents for NDS® (TSANDS) passes requests and commands between the SMDR and the eDirectory database and prepares the data for SME. In order to back up, TSANDS.NLM must be loaded on a server in the eDirectory tree.

- The storage device interface communicates between SME and the storage devices.

- Device drivers control the behavior of the storage devices.

- TSAProxy registers the workstation with the host server.

IMPORTANT: It is good practice to perform a backup before an upgrade, or any other major directory operation. But no SMS-based backup/restore is required during an eDirectory upgrade.

Using Backup and Restore Services on Windows NT

For instructions on backing up and restoring eDirectory on a specific server rather than the entire tree, refer to and .

The Novell SMS architecture extends to Windows NT. Use any SMS-compliant backup/restore utility to perform backup and restore operations on your eDirectory database located on your NT server. If you want to back up the eDirectory on NT database to a NetWare server, the following components are essential for backing up and restoring:

- The Storage Management Engine (SME) provides the interface for a user to perform backup and restore operations. SMSENGN.EXE is a basic SME provided by Novell for Windows NT. If you are already using an SMS-compliant backup engine, you should continue to use it. SMSENGN creates backups to a set of files—a data file (.DAT) and an index file (.IDX).

- ◆ The Storage Management Data Requester (SMDR) communicates between the SME and Target Service Agent (TSA) software. The SME and TSA can be on two different computers or the same computer. If the SME and TSA are on different computers, then an SMDR is required on both machines.

  If you are using Novell SMS on a NetWare network, you should use the SMDR provided with this utility in your regular SMS backup procedures.

- ◆ Target Service Agents for NDS (TSANDS) passes requests between the SMDR and the eDirectory database and prepares the data for SME.

  When eDirectory is installed, the SMDR and TSANDS are set up by default as a Windows NT service that is always available. If it is not available (service can be verified in Settings > Control Panel > Services), start W32SMDR.EXE located in the NDS\SMS directory.

Using Backup and Restore Services on Linux or Solaris

The ndsbackup utility is a command line utility which allows you to archive and restore eDirectory objects to and from a single file called ndsbackupfile. The actions of the ndsbackup utility are controlled by the command line options. The command line option is a string of characters containing exactly one function letter (c, r, t, s, or x) and zero or more function modifiers (letters or digits), depending on the function letter used. The string contains no space characters. Function modifier arguments are listed on the command line in the same order as their corresponding function modifiers appear in the string.

To archive or restore eDirectory objects, you need to specify the fully distinguished name of a leaf object or a container to be archived, extracted, or listed. To archive the whole tree, specify the Tree object. You can also back up the eDirectory schema by specifying Schema as the eDirectory object.

The ndsbackup utility allows you to customize the backup process. You can choose specific eDirectory objects to exclude or include in the backup session. Whether you use Exclude or Include usually depends on the size of the data you want to back up, compared to the size you do not want to back up. By combining the Include and Exclude options, you can control what is backed up. To back up most of the eDirectory tree structure while omitting only a small part, use the Exclude option to omit the part you do not want to back up. Everything that you do not want specifically excluded is included. After you exclude part of the structure, you cannot include objects below that container.

The following sections provide information about backing up and restoring eDirectory objects on Linux or Solaris systems:

ndsbackup

ndsbackup c [fevwXR] [ndsbackupfile] [exclude-file] [Replica-server-name] [-a admin-user] [-I include-file]... [eDirectoryobject]

ndsbackup r [fevwXR] [ndsbackupfile] [exclude-file] [Replica-server-name] [-a admin-user] [-I include-file]... [eDirectoryobject]

ndsbackup t [fevXR] [ndsbackupfile] [exclude-file] [Replica-server-name] [-a admin-user] [-I include-file]... [eDirectoryobject]

ndsbackup x [fevwXR] [ndsbackupfile] [exclude-file] [Replica-server-name] [-a admin-user] [-I include-file]... [eDirectoryobject]

ndsbackup s [evXR] [exclude-file] [Replica-server-name] [-a admin-user] [-I include-file]... [eDirectoryobject]

Table 136 **ndsbackup Parameters**

ndsbackup Parameter	Description
-a	The fully distinguished name of the user with administrator rights to the objects being archived or restored.
f	File. Use the ndsbackupfile argument as the name of the ndsbackupfile. If it is omitted, or if the name of the ndsbackupfile is -, ndsbackup writes to the standard output or reads from the standard input, whichever is appropriate. ndsbackup can be used as the head or tail of a pipeline.
e	Error. Exit immediately with a exit status if an unexpected error occurs.

ndsbackup Parameter	Description
R	Replica server name/IP address. Use the option to archive or restore eDirectory objects using a server holding the replica of the eDirectory partition. If you omit the R option, the local server is used.
v	Verbose. Output the name of each eDirectory object preceded by the function letter. With the t function, v provides additional information about the ndsbackupfile entries.
w	What. Output the action to be taken and the name of the eDirectory object, then await the user's confirmation. If you enter y, the action is performed. If you enter any other key, the action is not performed. This function modifier cannot be used with the t function.
-I	Include. Open the include file containing a list of eDirectory objects, one per line, and treat it as if each eDirectory object appeared separately on the command line. If an eDirectory object is specified in both the exclude file and the include file (or on the command line), it will be included. Be careful of trailing white spaces.
X	Exclude. Use the exclude-file argument as a file containing a list of eDirectory objects to be excluded from the ndsbackupfile when using the functions c, x, s, or t. Multiple X arguments may be used, with one exclude file per argument. If an eDirectory object is specified in both the exclude file and the include file (or on the command line), it will be included. Be careful of trailing white spaces.
NDSobject	The fully distinguished name of a leaf object or a container to be archived (when the c or r functions are specified), extracted (x) or listed (t). The action applies to all of the objects and (recursively) subordinate objects of that container. To archive the whole tree, specify the Tree object. You can also back up the eDirectory schema by specifying Schema as the eDirectory object. To back up the entire tree along with the schema, specify Full Directory Backup. If you do not specify the eDirectory object to be backed up, ndsbackup uses the default Full Directory Backup option.

Creating the ndsbackupfile

You can use the Create function specified by the c parameter to create the ndsbackupfile into which eDirectory objects are to be archived. Writing begins at the beginning of the file instead of the end. If a specified ndsbackup file exists, it will be overwritten.

To create the ndsbackupfile:

1 Use the following syntax:

```
ndsbackup c [fevwXR] [ndsbackupfile] [exclude-file]
[Replica-server-name] [-a admin-user] [-I include-
file]... [NDSobject]
```

Replacing Existing Objects for Restore

You can use the Replace function specified by the r option to back up the named eDirectory objects in the specified ndsbackupfile. The eDirectory objects are appended to the specified ndsbackupfile, effectively replacing the existing objects for restore.

To replace existing objects for restore:

1 Use the following syntax:

```
ndsbackup r [fevwXR] [ndsbackupfile] [exclude-file]
[Replica-server-name] [-a admin-user] [-I include-
file]... [NDSobject]
```

Scanning eDirectory Objects

You can use the Scan function specified by the s option to scan the eDirectory objects in a tree.

To scan the eDirectory objects in a tree:

1 Use the following syntax:

```
ndsbackup s [eSvwXR] [exclude-file] [Replica-server-name]
[-a admin-user] [-I include-file]... [NDSobject]
```

Obtaining a List eDirectory Objects from the ndsbackupfile

You can use the Table of Contents function specified by the t option, to list the names of the specified eDirectory objects each time they occur in the ndsbackupfile. If no argument is given, the names of all eDirectory objects in the ndsbackupfile are listed.

To obtain a list of eDirectory objects from the ndsbackupfile:

1 Use the following syntax:

```
ndsbackup t [fevXR] [ndsbackupfile] [exclude-file]
[Replica-server-name] [-a admin-user] [-I include-
file]... [NDSobject]
```

Restoring eDirectory Objects to the eDirectory Tree

You can use the Restore function specified by the x option, to extract the named eDirectory objects from the ndsbackupfile and restore to the eDirectory tree. If a named eDirectory object matches a container whose contents have been written to the ndsbackupfile, this container is recursively extracted.

To restore eDirectory objects to the eDirectory tree:

1 Use the following syntax:

```
ndsbackup x [fevwXR] [ndsbackupfile] [exclude-file]
[Replica-server-name] [-a admin-user] [-I include-
file]... [NDSobject]
```

Examples

To archive eDirectory objects in the container abc_inc:

1 Enter the following command:

ndsbackup cvf ndsbackupfile .O=abc_inc

To archive all the eDirectory objects in a tree:

1 Enter the following command:

ndsbackup cvf ndsbackupfile *tree_name*

To archive the eDirectory schema:

1 Enter the following command:

ndsbackup cvf ndsbackupfile Schema

To restore eDirectory objects from ndsbackupfile to eDirectory in the container abc_inc:

1 Enter the following command:

ndsbackup xvf ndsbackupfile .O=abc_inc

13 Maintaining Novell eDirectory

For Novell® eDirectory™ to perform optimally, you need to maintain the directory through routine health check procedures and upgrading or replacing hardware.

This chapter covers the following maintenance topics:

- Performance

- Health Checks

- Hardware Replacements

- eDirectory Recovery

Improving eDirectory Performance

The most significant setting that affects eDirectory performance is the cache. In earlier versions of NDS®, you could specify a block cache limit to regulate the amount of memory that the directory used for the cache. The default was 8 MB RAM for cache.

With eDirectory 8.5 or above, you can specify a block cache limit and an entry cache limit. The block cache, available in earlier versions of NDS, caches only physical blocks from the database. The entry cache, a new feature in eDirectory 8.5, caches logical entries from the database. The caching of entries reduces the processing time required to instantiate entries in memory from the block cache.

Although there is some redundancy between the two caches, each cache is designed to boost performance for different operations. Block cache is most useful for update operations. Entry cache is most useful for operations that browse the eDirectory tree by reading through entries, such as name resolution.

Both block and entry caches are useful in improving query performance. Block cache speeds up index searching. Entry cache speeds up the retrieval of entries referenced from an index.

The defaults for eDirectory 8.5 are listed below:

- If the server you are installing eDirectory on does not have a replica, the default is a hard memory limit of 16 MB, with 8 MB for block cache and 8 MB for entry cache.

  For more information, refer to "Understanding the Hard Memory Limit" on page 434.

- If the server contains a replica, the default is a dynamically adjusting limit of 51% of available memory, with a minimum threshold of 8 MB and a maximum threshold of keeping 24 MB available.

  For more information, refer to "Understanding the Dynamically Adjusting Limit" on page 433.

Distributing Memory between Entry and Block Caches

With an entry cache and a block cache, the total available memory for caching is shared between the two caches. The default is an equal division. To maintain the amount of block cache available in earlier versions of NDS 8, you need to double the total cache size for eDirectory. If you use the cache to boost LDIF-import performance, for example, you can either double the total cache size or change the default cache settings. To change the default cache settings, refer to "Configuring Dynamically Adjusting and Hard Memory Limits" on page 435.

The more blocks and entries that can be cached, the better the overall performance will be. The ideal is to cache the entire database in both the entry and block caches, although this is not possible for extremely large databases. Generally, you should try to get as close to a 1:1 ratio of block cache to DIB Set as possible. For entry cache, you should try to get as close to a 1:2 or 1:4 ratio. For the best performance, exceed these ratios.

Using the Default Cache Settings

eDirectory 8.5 provides two methods for controlling cache memory consumption: a dynamically adjusting limit and a hard memory limit. You can use either method, but you cannot use them at the same time because they are mutually exclusive. The last method used always replaces any prior settings.

Understanding the Dynamically Adjusting Limit

The dynamically adjusting limit causes eDirectory to periodically adjust its memory consumption in response to the ebb and flow of memory consumption by other processes. You specify the limit as a percentage of available physical memory. Using this percentage, eDirectory recalculates a new memory limit at fixed intervals. The new memory limit is the percentage of physical memory available at the time.

Along with the percentage, you can set a maximum and minimum threshold. The threshold is the number of bytes that eDirectory will adjust to. It can be set as either the number of bytes to use or the number of bytes to leave available. The minimum threshold default is 16 MB. The maximum threshold default is 4 GB.

If the minimum and maximum threshold limits are not compatible, the minimum threshold limit is followed. For example, specify the following settings:

Minimum threshold	8 MB
Percentage of available physical memory to use	75
Maximum threshold	Keep 10 MB available

When eDirectory adjusts its cache limit, there is 16 MB of available physical memory. eDirectory calculates a new limit of 12 MB. eDirectory checks to see whether the new limit falls within the range of minimum and maximum thresholds. In this example, the maximum threshold requires that 10 MB must remain available, so eDirectory sets the limit to 6 MB. However, the minimum threshold is 8 MB, so eDirectory sets the final limit to 8 MB.

With the dynamically adjusting limit, you also specify the interval length. The default interval is 15 seconds. The shorter the interval, the more the memory consumption is based on current conditions. However, shorter intervals are not necessarily better because the percentage recalculation will create more memory allocation and freeing.

Understanding the Hard Memory Limit

The hard memory limit is the method that earlier versions of eDirectory use to regulate memory consumption. You set a hard memory limit in one of the following ways:

- ◆ Fixed number of bytes

- ◆ Percentage of physical memory

   The percentage of physical memory at the interval becomes a fixed number of bytes.

- ◆ Percentage of available physical memory

   The percentage of available physical memory at the interval becomes a fixed number of bytes.

Cleaning Up the Cache

NDS 8 creates multiple versions of blocks and entries in its cache for transaction integrity purposes. Earlier versions of NDS 8 did not remove these blocks and entries when they were no longer needed. In eDirectory 8.5, a background process periodically browses the cache and cleans out older versions. This helps minimize cache memory consumption. The default browsing interval is 15 seconds.

Configuring Dynamically Adjusting and Hard Memory Limits

1 Open _NDSDB.INI in a text editor.

On NetWare®, this file is in SYS:\NETWARE. On Windows* NT* and Windows 2000, this file is generally in \NOVELL\NDS\DIBFILES.

2 Add the applicable syntax to the file:

Table 137

Command	Variable Explanation	Definition
cache=*cache_bytes*	Fixed number of bytes you want used.	Sets a hard memory limit.
		For example, to set a hard limit of 8 MB, type `cache=8000000`.
cache=*cache_options*	Multiple options can be specified in any order, separated by commas.	Sets a hard memory or dynamically adjusting limit.
	◆ DYN	For example, to set a dynamically adjusting limit of 75% of available memory and a minimum of 16 MB, type `cache=DYN, %:75,MIN:16000000`.
	Sets a dynamically adjusting limit.	
	◆ HARD	
	Sets a hard memory limit.	Or to set a hard limit of 75% of total physical memory and a minimum of 16 MB, type `cache=HARD,%:75,MIN: 16000000`.
	◆ %:*percentage*	
	Percentage of available or physical memory to use.	
	◆ AVAIL or TOTAL	
	Percentage of available or total physical memory for hard memory limit only.	
	◆ MIN:*number_of_bytes*	
	Minimum number of bytes.	
	◆ MAX:*number_of_bytes*	
	Maximum number of bytes.	
	◆ LEAVE:*number_of_bytes*	
	Minimum number of bytes to leave.	

3 (Optional) To specify the dynamic adjusting limit interval, add the following line:

`cacheadjustinterval=number_of_seconds`

4 (Optional) To specify the interval for cleaning up older versions of entries and blocks, add the following line:

`cachecleanupinterval=number_of_seconds`

5 (Optional) To change the percentage split between block cache and entry cache, add the following line:

`blockcachepercent=percent`

The variable *percent* should be between 0 and 100. The percentage you specify is the percentage of cache memory used for the block cache. The remaining percentage is used for the entry cache. We do not recommend setting the percentage to 0.

6 Restart the eDirectory server for the changes to take effect.

Configuring Limits Using DSTRACE

If you are using eDirectory for NetWare, you can configure the dynamically adjusting and hard memory limits in DSTRACE. You do not need to restart the server for the changes to take effect.

1 (Optional) To set a fixed hard limit, type the following at the server console:

`SET DSTRACE=!MBamount_of_RAM_to_use_in_bytes`

For example, if you want to set a hard limit of 8 MB, type:

```
SET DSTRACE=!MB8388608
```

2 (Optional) To set a calculated hard limit, type the following at the server console. Only type the options you want to specify.

`SET DSTRACE=!MHARD,AVAIL OR`
`  TOTAL,%:percent,MIN:number_of_bytes,MAX:number_of_`
`  bytes,LEAVE:number_of_bytes_to_leave,NOSAVE`

For example, to set a hard limit of 75% of total physical memory and minimum of 16 MB, and to specify not to save these options to the startup file, type:

```
SET DSTRACE=!MHARD,%:75,MIN:16777216,NOSAVE
```

3 (Optional) To set a dynamically adjusting limit, type the following at the server console:

`SET DSTRACE=!MDYN,%:percent,MIN:number_of_bytes,MAX:`
`  number_of_bytes,LEAVE:number_of_bytes_to_leave,`
`  NOSAVE`

For example, to set a dynamic limit of 75% of available memory and a minimum of 8 GB, type:

```
SET DSTRACE=!MDYN,%:75,MIN:8388608
```

Improving eDirectory Performance on Linux and Solaris Systems

The following sections provide information about how you can improve the performance of eDirectory on UNIX* systems:

Fine-Tuning the eDirectory Server

Novell eDirectory on Linux* and Solaris* uses a dynamically adjusted thread pool to service client requests. The thread pool is self-adjusting and delivers optimum performance in most cases. However, you can avoid the delay caused by starting up threads when there is a sudden load on the server by setting the following parameters in the /etc/nds.conf file.

Table 138

Parameter	Description
n4u.server.max-threads	Absolute maximum number of threads
n4u.server.idle-threads	Number of threads to be kept idle
n4u.server.start-threads	Number of threads to be pre-started

Set the value for the n4u.server.max-threads parameter based on the maximum number of simultaneous clients that need to be serviced. Novell eDirectory internally needs about 16 threads for regular usage. You can add one thread for every 255 LDAP connections to monitor LDAP connections. Add an additional thread for every four clients that need to be serviced simultaneously. Set the value for the n4u.sserver.idle-threads and n4u.server.start-threads parameters based on the average client load.

A number of LDAP Server objects can be members of one LDAP group. All the LDAP servers share the properties of the LDAP Group object (for example, class and attribute mappings, proxy user, and so on). Therefore, if you have a list of your own class and attribute mappings to add, you can add them in one group and make all the servers a member of this group.

Optimizing eDirectory Cache

eDirectory uses persistent caching so that changes being made to a server are held in a vector. If the server crashes in the middle of changes, eDirectory will load faster and synchronize the changes in seconds when the server is brought back up. Novell eDirectory uses a rollback model with a log file to roll forward transactions in the event of a system failure.

Novell eDirectory uses approximately 50% of available free memory for the cache, leaving at least 24 MB for the OS. This algorithm is used only if the host OS supports the call that enables you to determine the amount of free memory available. Although this algorithm works well for Windows and NetWare, it does not work for UNIX systems. On UNIX systems, the free available memory reported by the OS will be less than other operating systems because of the way the UNIX OS uses free memory for internal caching of file system blocks, frequently run programs, libraries, and so on. In addition to this memory allocation, libraries on UNIX normally do not return the freed memory back to the OS.

For these reasons, we recommend allocating a fixed amount of RAM to the cache. To do this, create a file called _ndsdb.ini in the DIB Set directory (/var/nds/dib by default), then specify a value for the cache parameter in the file. Novell eDirectory internally allocates this cache equally between the block cache and record cache. The cache parameter can be set to either an absolute value or the following set of parameters separated by commas:

Table 139

eDirectory Cache Parameters	Description
dyn	Specifies that dynamic cache tuning is in effect (default)
hard	Specifies a hard limit
avail	Specifies the percentage of available memory to use

eDirectory Cache Parameters	Description
total	Specifies the percentage of total memory to use
%:*percentage*	Specifies the percentage ratio to use
leave:*bytes*	Specifies the minimum number of bytes to leave
min:*bytes*	Specifies the minimum cache size in bytes
max:*bytes*	Specifies the maximum cache size in bytes

According to the algorithm, the default setting for Novell eDirectory is the following:

```
cache=dyn,avail,%:50,min:8388508,max:4294967295,leave:2516582
4
```

This indicates the following:

 * The available memory must be used for calculation.

 * The minimum cache size is 8 MB.

 * There is no maximum limit.

 * Up to 50% of available memory will be used.

 * 24 MB should be left for the OS.

   eDirectory operates with a hard limit of 16 MB, so that all applications are started and the system is stabilized.

You can also configure Novell eDirectory to use a percentage of the total memory. To do so, specify the cache as shown below:

`cache=hard,total,%:percentage_of_total_memory_in_bytes`

Optimizing Bulkload Data

By default, eDirectory uses dynamic cache. If you have sufficient RAM to increase the eDirectory cache size, you can increase the performance of eDirectory considerably for large databases by allocating more RAM to the eDirectory cache. For more information, see . Performance of bulkload using the Import/Export utility is affected considerably by cache size. (A larger cache size is faster.)

Cache size is set using the following command at the ndstrace command line:

```
set ndstrace = !m[hexadecimal KB]
```

or

```
set ndstrace = !mb[bytes]
```

For example, the set ndstrace=!m4F00 command allocates approximately 20 MB of RAM to the eDirectory cache. If eDirectory is the server's only application, you can set the eDirectory cache up to 80% of the total memory.

IMPORTANT: You should avoid setting the cache memory size above 40% of the total memory if the server is hosting services or applications other than eDirectory.

The smallest tested cache size is 0 and the largest is 2 GB. Determining the proper cache size depends on the memory needs of other processes running on the same server, and on the amount of disk cache required. You should test a variety of cache sizes to find a good balance. If eDirectory is essentially the only application, give it as much cache as possible. All allocated cache will eventually be used. eDirectory performance on highly volatile data is improved with more cache.

To optimize the bulkload performance, allocate a higher percentage of the Novell eDirectory cache for block cache. We recommend setting a value of 80% for block cache. To do so, modify the value for the blockcachepercentage parameter in the _ndsdb.ini file located in the /var/nds directory. The cache parameter must be set before you specify a value for the blockcachepercentage parameter.

Optimizing LBURP Transaction Size

The LBURP transaction size determines the number of records that will be sent from the Novell Import/Export client to the LDAP server in a single LBURP packet. You can increase the transaction size to ensure that multiple add operations can be performed in a single request. The transaction size can be modified by specifying the required value for the n4u.ldap.lburp.transize parameter in /etc/nds.conf. The default transaction size is 25. The default value is appropriate for small LDIF files, but not for large number of records. You can provide a transaction size in the hard-limit range of 1 and 250.

In ideal scenarios, a higher transaction size ensures faster performance. However, the transaction size must not be set to arbitrarily high values for the following reasons:

- A larger transaction size requires the server to allocate more memory to process the transaction. If the system is running low on memory, this can cause a slowdown due to swapping.

- Even if a single error exists in the transaction, (including cases where the object to be added already exists in the directory), the LBURP optimization will be futile, since the bulkload operation will begin to add objects to eDirectory individually, which may cause a performance problem. Ensure that the LDIF file is free of errors and that any entries already existing in eDirectory have been commented out.

  See "Debugging LDIF Files" on page 487 for more information.

- LBURP optimization works only for leaf objects. However, the optimization is lost within a transaction if the transaction contains both container and its subordinate objects. Because of this, we recommend that you enable the use of forward references.

  See "Enabling Forward References" on page 488 for more information.

Tuning the Solaris OS for Novell eDirectory

Ensure that you have applied all the recommended patches to the Solaris OS. For more information, see "Solaris" on page 30.

The following sections provide information about how can tune the Solaris kernel, network, and file system:

- "Tuning the Solaris Kernel" on page 441
- "Tuning the Solaris Network" on page 442
- "Tuning the Solaris File System" on page 442

Tuning the Solaris Kernel

You can set the following kernel variables in the /etc/system file on the Solaris system to optimize the performance of Novell eDirectory:

```
set priority_paging=1
set maxphys=1048576
set md_maxphys=1048576
set ufs:ufs_LW=1/128_of_available_memory
set ufs:ufs_HW=1/64_of_available_memory
set tcp:tcp_conn_hash_size=8192 (This can be increased to
262144 based on the number of LDAP clients.)
```

Tuning the Solaris Network

You can enhance LDAP search performance using the Solaris ndd command, which allows you to analyze and modify tunable parameters that affect networking operation and behavior. Use the following syntax to do so:

```
ndd -set /dev/tcp variable_name variable_value
```

The recommended values for the variables that you can set are provided below:

```
tcp_conn_req_max_q: 1024
tcp_close_wait_interval: 60000
tcp_xmit_hiwat: 32768
tcp_xmit_lowat: 32768
tcp_slow_start_initial: 2
```

Tuning the Solaris File System

Novell eDirectory performance on Solaris can be improved if the Solaris file system is adequately tuned, especially for bulk loading data into the directory. File system tuning for Novell eDirectory is similar to tuning for a database. See the Sunworld* Web site (http://www.sunworld.com/sunworldonline/) for more information on the Solaris file system.

Keeping eDirectory Healthy

The health of directory services is vital to any organization. Regular health checks will keep your directory running smoothly and will make upgrades and troubleshooting much easier.

When to Perform Health Checks

In general, if your network doesn't change often (servers and partitions are only added every couple of months; only simple changes are made frequently), perform health checks once a month.

If your network is more dynamic (partitions or servers are added weekly, your organization is reorganizing), perform health checks weekly.

Adjust the frequency of health checks as your environment changes. Factors that influence the timing of your health checks include the

- Number of partitions and replicas
- Stability of replica holding servers
- Amount of information in an eDirectory partition
- NDS object size
- Number of errors in previous DSRepairs

Health Check Overview

A complete health check includes checking

- NDS version

  Running different versions of NDS on the same version of NetWare can cause synchronization problems. If your version of NDS is outdated, download the latest software patch from NDS Patches and Files (http://www.support.novell.com/filefinder/5069/index.html)

- Time synchronization

  All NDS servers must maintain accurate time. Time stamps are assigned to each object and property. They ensure the correct order for object and property updates. Using time stamps, NDS determines which replicas need to be synchronized.

- Partition continuity

  Partition continuity ensures that all replicas for a partition can be updated with directory changes.

- Background processes: NDS changes are replicated in background processes.

  - External references
  - Obituaries
  - Remote server IDs
  - Unknown NDS objects
  - The NDS Schema

- NDS SET parameters

For step-by-step instructions to complete these checks, see the appropriate section for your operating system:

Maintaining eDirectory on NetWare

To maintain eDirectory, perform the following operations for every NetWare server. Perform Step 9 on page 446 after business hours and only when errors occur during Step 1 through Step 10 on page 449.

You need to perform all ten steps for every server, even if your tree is very large with many partitions. However, if you must abbreviate maintenance, perform all ten steps on servers holding the master replica for each partition, starting with the Master replica server for the Tree partition and working down the tree.

1 Check the version of DS.NLM.

DS.NLM should be the same version on every NetWare 4.1*x* and NetWare 5.*x* server in the tree.

You can view the version of DS.NLM for each server known to the server you are using by performing the time synchronization in Step 2.

2 In DSREPAIR, click Available Options > Advanced Options > Time Synchronization to update time synchronization.

Time synchronization is critical for directory services functions.

3 Display server-to-server synchronization.

3a Type the following from the server console:

◆ **SET DSTRACE=ON**

This activates the trace screen for directory services transactions.

◆ **SET DSTRACE=+S**

This filter lets you view the synchronization of objects.

◆ **SET DSTRACE=*H**

This initiates synchronization between servers.

To view the directory services trace screen, select Directory Services from the Current Screens list > press Ctrl+Esc. If there are no errors, a line displays `All processed = YES`. This message is displayed for each partition contained on this server.

A server must have a replica to display any directory services trace information.

3b If the information cannot fit on a single screen, use the following commands:

- **SET TTF=ON**

  This sends the DSTRACE screen to the SYS:SYSTEM\DSTRACE.DBG file.

- **SET DSTRACE=*R**

  This resets the file to 0 bytes, deleting previous entries.

- **SET TTF=OFF**

  This is done once eDirectory completes synchronizing all partitions.

  Map a drive to SYS:SYSTEM > open the DSTRACE.DBG file in a text editor.

  Search for -6, which shows any eDirectory errors during synchronization, such as `-625`.

  or

  Search for Yes, which shows successful synchronization for a partition.

4 In DSREPAIR, click Available Options > Advanced Options > Report Synchronization Status to report replica synchronization.

A server must have a replica for this operation to display replica synchronization status.

5 In DSREPAIR, click Available Options > Advanced Options > Check External References to check external references.

This option displays external references and obituaries and shows you the states of all servers in the back link list for the obituaries.

6 Check the replica state.

6a In DSREPAIR, click Available Options > Advanced Options > Replica and Partition Operations.

6b Verify that the replica state is On.

7 Check the replica ring.

7a Open DSREPAIR on the server holding the master replica of each partition and one of the servers holding a read/write replica to check for replica ring mismatches.

7b Click Available Options > Advanced Options > Replica and Partition Operations > View Replica Ring.

7c Verify that the servers holding replicas of that partition are correct.

8 To check the schema, type the following at the server console:

- **SET DSTRACE=ON**

  This activates the trace screen for directory services transactions.

- **SET DSTRACE=+SCHEMA**

  This displays schema information.

- **SET DSTRACE=*SS**

  This initiates schema synchronization.

To view the directory services trace screen, select Directory Services from the Current Screens list > press Ctrl+Esc. Check for the following message:

SCHEMA: All Processed = YES

A server must have a replica to display any directory services trace information.

9 Repair the local database.

You might want to perform this task after hours.

9a In DSREPAIR, click Available Options > Advanced Options > Repair Local DS Database.

9b Check Yes on Check Local References and Rebuild Operational Schema.

All of the other options on this page can be checked No.

This option locks the directory services database. DSREPAIR displays a message stating that authentication cannot occur with this server when directory services is locked. (Users are not be able to log in to this server.) For this reason, this operation may need to be performed after business hours.

DSTRACE, if left running, requires server resources. After completing the DSTRACE checks, enter the following DSTRACE commands to turn it off:

```
Set DSTRACE=nodebug
Set DSTRACE=+min
Set DSTRACE=off
```

Maintaining eDirectory on Windows NT/2000

To maintain eDirectory, perform the following operations each week for every Windows NT/2000 server. Perform after business hours and only when errors occur during through .

For very large trees and for a large number of partitions, you still need to perform all ten steps for every server, but for an abbreviated version, perform all ten steps on servers holding the master replica for each partition, starting with the master replica server for the Tree partition and working down the tree.

1 Check the version of DS.DLM.

DS.DLM should be the same version on every NT 3.51 and NT 4 server in the tree.

You can view the version of DS.DLM for each server known to the server you are using by performing the time synchronization in .

2 Update time synchronization.

Time synchronization is critical for directory services functions.

2a Go to the NDSCONSOLE > select DSREPAIR> click Start.

2b Click Repair > Time Synchronization.

3 Display server-to-server synchronization.

A server must have a replica to display any eDirectory server trace information.

3a Go to the NDSCONSOLE > select DSTRACE.DLM > click Start.

3b When the NDS Server Trace Utility window appears, click Edit > Options.

> **HINT:** Keep this window open. You will be referring back to it numerous times during the next few steps.

3c Check the Replication Process check box > click OK.

3d Go to the NDSCONSOLE > select DS.DLM > click Configuration.

3e When the NDS Configuration dialog box appears, click the Triggers tab > click Replica Sync.

3f Refer back to the NDS Server Trace Utility screen and look for this message: `All processed = YES.`

If the information does not fit on the screen, or if you want to save this information so you can review it later, follow these directions to create a log file for ease of viewing and storing the information:

◆ Return to the NDS Server Trace Utility window > click File > click New.

◆ Once a new file is created, you can save all the DSTRACE messages in this file, which can be reviewed at a later date.

4 In DSREPAIR, click Repair > Report Synchronization to report replica synchronization.

A server must have a replica for this operation to display replica synchronization status.

5 In DSREPAIR > click Repair > Check External References to check external references.

This option displays external references and obituaries and shows you the states of all servers in the back link list for the obituaries.

6 Check the replica state.

6a In DSREPAIR, select a partition in the tree view.

When you select a partition, the replica state of the selected partition will be listed in the list view on the right.

6b Verify that the replica state is On.

7 Check the replica ring.

7a Open DSREPAIR on the server holding the master replica of each partition and one of the servers holding a read/write replica to check for replica ring mismatches.

7b In DSREPAIR, select and expand a partition in the tree view, then expand a replica.

You should now be able to see all the servers in the replica ring.

7c Verify that the servers holding replicas of that partition are correct.

8 Check the schema.

A server must have a replica to display any eDirectory server trace information.

8a Return to the NDS Server Trace Utility window.

If you've closed the window, go to NDSCONSOLE > select DSTRACE.DLM > click Start.

8b In the NDS Server Trace Utility window, click Edit > click Options > check the Schema check box > click OK.

8c Return to the NDSCONSOLE > select DS.DLM > click Configuration.

8d In the NDS Configuration dialog box > click the Triggers tab > click Schema Sync.

8e Return to the NDS Server Trace Utility window and look for this message: SCHEMA: All Processed = YES. When this message appears, click OK in the NDS Configuration dialog box.

9 Repair the local database.

You might want to perform this after hours.

9a In DSREPAIR > click Repair > Local Database Repair.

9b Check the Check Local References and Rebuild Operational Schema check boxes and next to Check Local References and Rebuild Operational Schema > uncheck all other options > click Repair > click Yes.

This option locks the directory services database. DSREPAIR displays a message stating that authentication cannot occur with this server when directory services is locked (users are not be able to log in to this server). For this reason, this operation may need to be performed after business hours.

10 To turn DSTRACE off, go to NDSCONSOLE > select File > select Exit.

DSTRACE, if left running, requires server resources. After completing the DSTRACE checks, you should turn DSTRACE off.

Maintaining Novell eDirectory on Linux and Solaris

To maintain eDirectory, perform the following operations each week for every Linux or Solaris system. Perform Step 8 on page 452 after business hours and only when errors occur during Step 1 through Step 9 on page 452.

For large trees and for a large number of partitions, you still need to perform all ten steps for every server, but for an abbreviated version, perform all ten steps on servers holding the master replica for each partition, starting with the master replica server for the Tree partition and working down the tree.

1 Load ndsrepair > enter **ndsrepair -T** at the terminal to check the version of the eDirectory daemon (ndsd).

   The eDirectory daemon should be the same version on every UNIX server in the tree.

2 Display server-to-server synchronization.

   A server must have a replica to display any eDirectory server trace information.

   2a Start the ndstrace utility.

   HINT: Keep this utility open. You will be referring back to it numerous times during the next few steps.

   2b Enter the following command to enable replica synchronization messages:

   dstrace SKLK

   2c Enter the following command to initiate an immediate replica synchronization:

   set dstrace=*H

   2d Check the ndstrace messages and look for this message: All processed = YES.

3 If the ndstrace messages do not fit on the screen, or if you want to save this information so you can review it later, create a log file for ease of viewing and storing the information by following these directions:

   3a Enter the following command to enable ndstrace messages to be logged to the file:

   dstrace file on

   3b If you want to reset the log file size to zero, enter the following command at the ndstrace command line:

   set dstrace =*R

3c After the ndstrace messages are logged to a file, enter the following command to disable the messages from logging to a file:

```
dstrace file off
```

When a new file is created, you can save all the ndstrace messages in this file, which can be reviewed at a later date.

4 Report replica synchronization.

A server must have a replica for this operation to display replica synchronization status.

4a Enter the following command:

```
ndsrepair -P
```

4b Select the partition and select the option that specifies the Report Synchronization Status of All Servers operation.

5 Enter the following command to check external references.

```
ndsrepair -C
```

This option displays external references and obituaries and shows you the states of all servers in the back link list for the obituaries.

6 Check the replica state and replica ring.

6a Enter the following command:

```
ndsrepair -P
```

6b Select the partition and select the option that specifies the View Replica Ring and Check Replica State. Ensure that the replica state is On.

7 Check the schema.

A server must have a replica to display any eDirectory server trace information.

7a Enter the following command to enable schema replicate messages:

```
dstrace scma
```

7b Enter the following command to start schema synchronization:

```
set dstrace=*SS
```

7c Check the ndstrace message and look for this message: `Schema: All Processed = Yes.`

8 Repair the local database.

You might want to perform this after hours.

8a Enter the following command:

ndsrepair -R

8b Enable the Rebuild Operation Schema (-o) and Check Local Reference (-c) suboptions of the ndsrepair utility.

These options lock the directory services database. NDSREPAIR displays a message stating that authentication cannot occur with this server when directory services is locked. Users are not be able to log in to this server. For this reason, this operation may need to be performed after business hours.

9 Enter the **exit** command to exit NDSTRACE.

For More Information

The tools and techniques used to keep eDirectory healthy are documented in the new Novell Certified Directory Engineer Course 991: Advanced NDS Tools and Diagnostics. In this course you learn how to:

- ◆ Perform eDirectory health checks
- ◆ Perform eDirectory operations properly
- ◆ Properly diagnose, troubleshoot, and resolve eDirectory issues
- ◆ Use eDirectory troubleshooting tools and utilities

To learn more about this course, visit the Novell Education Web site (http://education.novell.com).

Novell Consulting Services also provides eDirectory health checks for customers. For more information, visit the Novell Customer Services Web site (http://services.novell.com).

Monitoring

The Novell DSTRACE utility runs on NetWare, Windows NT, Linux, and Solaris. This tool helps you monitor the vast resources of eDirectory. You can also invest in third-party products that provide additional management solutions for your eDirectory environment. For more information, see the following Web sites:

- BindView (http://www.bindview.com)
- Blue Lance (http://www.bluelance.com)
- NetPro* (http://www.netpro.com)

If you need to monitor or audit certain characteristics of eDirectory that our partners do not provide, Novell Consulting Services can use the Novell Event System for customized assessment and auditing.

Upgrading/Replacing Hardware on NetWare

Instructions in the following section give information you need to know about eDirectory on a specific server when you upgrade or replace hardware. For information on backing up and restoring eDirectory on NetWare on an entire tree, refer to "Using Backup and Restore Services on NetWare" on page 423.

Options available in NWCONFIG.NLM on NetWare allow you to prepare eDirectory information on a server for a planned hardware upgrade, hard drive upgrade, or replacement of the server. The following instructions are designed for situations where a server is actually replaced. For naming purposes, the old server is referred to as Server A and its replacement or the new server is referred to as Server B.

To prepare for the upgrade, the eDirectory backup utility creates backup files in the directory SYS:\SYSTEM\$HWNDS.BAK named *.$HW that store all of the eDirectory information on a server, including replica information. It also locks and disables eDirectory on this server, preventing any data change. To other servers that normally communicate with this server, the server appears to be down. Any eDirectory information that is normally sent to the locked server is stored by other servers in the tree. The stored information is used to synchronize the server when it comes back online.

Because other servers in the eDirectory tree expect the server to come back online quickly, you should complete the upgrade promptly and restore eDirectory information on the server as soon as possible.

The Restore Local DS Information after Hardware Upgrade option uses the backup to restore eDirectory information on this server.

Figure 39 on page 454 shows the procedure for replacing hardware:

Figure 39   Replacing a Server
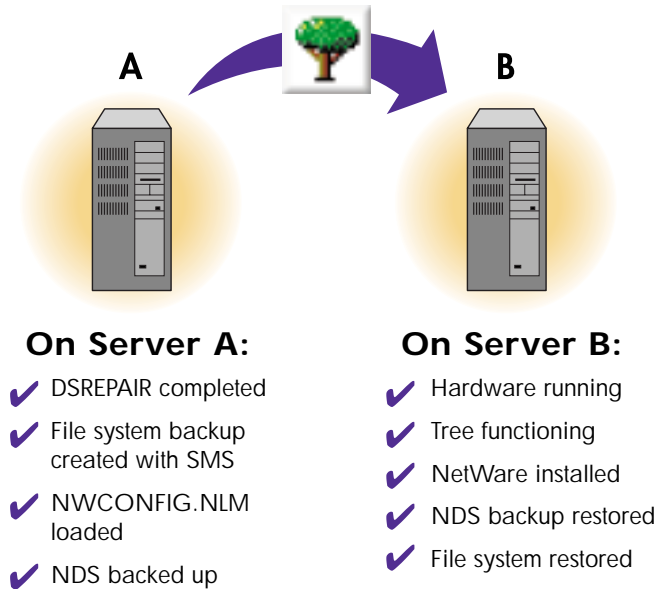
On Server A:
✔ DSREPAIR completed
✔ File system backup created with SMS
✔ NWCONFIG.NLM loaded
✔ NDS backed up

On Server B:
✔ Hardware running
✔ Tree functioning
✔ NetWare installed
✔ NDS backup restored
✔ File system restored

Preparing for a Hardware Change

Use the following checklist to determine if you are ready to start this procedure.

❑ Make sure the tree for Server A is healthy by running DSREPAIR on the server that holds the master of the Tree partition and by running time synchronization.

❑ Run DSREPAIR on the database of Server A.

❑ Make a backup of Server A's file system. Use SMS to make the backup of the file system. For information on using SMS, see Chapter 12, "Backing Up and Restoring Novell eDirectory," on page 419. This step ensures that your trustees will not be lost.

❑ Make sure that Server A has the latest version of NetWare installed.

❑ Make sure that Server B is up and the latest version of NetWare is running.

❑ Install Server B in its own tree.

Creating a Backup of eDirectory

The eDirectory backup utility creates backup files with the extension *.HW in the SYS:\SYSTEM\$HWNDS.BAK directory to store all the eDirectory information on the server. Use the following instructions to create a backup of eDirectory prior to a hardware upgrade:

1 At the server console, load NWCONFIG.NLM on Server A.

2 Enter **NWCONFIG**.

3 Select Directory Options > Directory Backup and Restore Options.

4 Select Save Local DS Information Prior to Hardware Upgrade > press Enter.

A help screen appears with eDirectory backup information. Press Enter to continue after reading this screen.

5 Enter the administrator name and password.

The system logs you in to eDirectory and creates the backup files.

6 Save the backup file to a location you know you can access.

You can save to SYS:SYSTEM on Server B but only if it is up and running.

7 Exit NWCONFIG.NLM and bring down Server A by entering **down**.

Server A's eDirectory database is now locked. Complete the upgrade promptly and restore eDirectory information on Server B as soon as possible.

Restoring eDirectory Information after a Hardware Upgrade

The Restore DS Information after a Hardware Upgrade option uses the files created during the backup to restore eDirectory information on Server B. Before eDirectory is restored, this utility ensures that the server is in the same relative state as before the upgrade. NWCONFIG.NLM ensures that the server's object and authentication keys still exist and that the server still exists in all the replica rings for copies that were on this server before the upgrade.

1 Rename Server B using Server A's tree name, address, and server name in AUTOEXEC.NCF.

2 Enter **NWCONFIG** at the server console prompt on Server B.

3 Select Directory Options > Directory Backup and Restore Options.

4 Select Restore Local DS Information after Hardware Upgrade > press Enter.

5 Verify the path to the backup files or press F3 to enter a new path.

6 Use SMS to restore the file system backup you made of Server A that restores the file system and trustees.

7 Use ConsoleOne™ to check the server. Make sure that login scripts and printing work correctly.

If Server B does not work correctly and you need to make Server A functional immediately do the following:

1 Unplug Server B's network cable or down the server.

2 Reattach Server A to the network and start it.

Ignore system messages requesting you to run DSREPAIR.

3 Load NWCONFIG.NLM on Server A.

4 Select Directory Options > Directory Backup and Restore Options.

5 Select Restore Local DS Information after Hardware Upgrade.

6 Enter the path to backup files on Server A.

This unlocks the eDirectory on the server and returns it to its state before the upgrade.

7 Remove eDirectory from Server B and try the upgrade again.

Upgrading/Replacing Hardware on Windows NT/2000

Instructions in the following section give information you need to know about eDirectory on a specific server when you upgrade or replace hardware. For information on backing up and restoring eDirectory on Windows NT/2000 on an entire tree, refer to .

Options available in INSTALL.DLM allow you to prepare eDirectory information on a server for a planned hardware upgrade, hard drive upgrade, or replacement of the server. The following instructions are designed for situations where a server is actually replaced. For naming purposes, the old server is referred to as Server A and its replacement or the new server is referred to as Server B.

To prepare for the upgrade, the NT/2000 backup utility creates a backup file named BACKUP.NDS that stores all of the eDirectory information on a server, including replica information. It also locks and disables eDirectory on this server, preventing any data change. To other servers that normally communicate with this server, the server appears to be down. Any eDirectory information that is normally sent to the locked server is stored by other servers in the tree. The stored information is used to synchronize the server when it comes back online.

Because other servers in the eDirectory tree expect the server to come back online quickly, you should complete the upgrade promptly and restore eDirectory information on the server as soon as possible.

The Restore Local DS Information after Hardware Upgrade option uses the backup to restore eDirectory information on this server.

Figure 40 shows the procedure for replacing hardware.

Figure 40   Replacing a Server

On Server A:
✔ DSREPAIR completed
✔ Create file system backup
✔ INSTALL.DLM loaded
✔ NDS backed up

On Server B:
✔ Hardware running
✔ Tree functioning
✔ NT Server installed
✔ NDS backup restored
✔ File system restored

Preparing for a Hardware Change

Use the following checklist to determine if you are ready to start this procedure.

❑ Ensure that the tree for Server A is healthy by running DSREPAIR on the server that holds the master of the Tree partition and by running time synchronization.

❑ Run DSREPAIR on the database of Server A.

❑ Make a backup of Server A's file system. Use the NT backup utility to make the backup of the file system.

❑ Make sure that Server A has the latest version of NT installed.

❑ Make sure that Server B is up and the latest version of NT is running.

❑ Install the latest version of Account Management on the NT server.

❑ Install Server B in its own tree.

Creating a Backup of eDirectory

The eDirectory backup utility creates a backup file named $HWNDS.BAK in C:\NOVELL\NDS\DIBFILES directory to store all the eDirectory information on the server. Use the following instructions to prepare eDirectory for a hardware upgrade:

1 At the server console, run NDSCONS.EXE > click INSTALL.DLM > click Start.

2 Click Save Local DS Information Prior to Hardware Upgrade > press Next.

3 Enter the administrator name and password.

4 Enter the destination path for the backup file > click Finish.

   The system creates the backup file in the same directory as the eDirectory database files, C:\NOVELL\NDS\DIBFILES\$HWNDS.BAK.

Server A's DS database is now locked. Complete the upgrade promptly and restore eDirectory information on Server B as soon as possible.

Restoring eDirectory Information after a Hardware Upgrade

The Restore DS Information after a Hardware Upgrade option uses the files created during the backup to restore eDirectory information on Server B. Before eDirectory is restored, this utility ensures that the server is in the same relative state as before the upgrade. The INSTALL.DLM ensures that the server's object and authentication keys still exist and that the server still exists in all the replica rings for copies that were on this server before the upgrade.

1 In the eDirectory console click INSTALL.DLM > click Start.

2 Click Restore Local DS Information after Hardware Upgrade and press Next.

3 Verify the path to the C:\NOVELL\NDS\DIBFILES directory or enter a new path > click Finish.

4 Use your NT backup services to restore the file system backup you made of Server A and its trustees.

5 Make sure that login scripts and printing work correctly.

If Server B does not work correctly and you need to make Server A functional immediately, do the following:

1 Unplug Server B's network cable or down the server.

2 Reattach Server A to the network and start it.

Ignore system messages requesting you to run DSREPAIR.

3 Click INSTALL.DLM > click Start.

4 Click Restore Local DS Information after Hardware Upgrade > click Next.

5 Verify the path to the C:\NOVELL\NDS\DIBFILES directory or enter a new path > click Finish.

6 When restore is complete, click Done.

This unlocks the eDirectory on the server and returns it to its state before the upgrade.

7 Remove eDirectory from Server B and try the upgrade again.

Upgrading/Replacing Hardware on Linux and Solaris

The following sections provide information about manually upgrading or replacing hardware on Linux or Solaris systems. For naming purposes, the old server is referred to as Server A and its replacement or the new server is referred to as Server B.

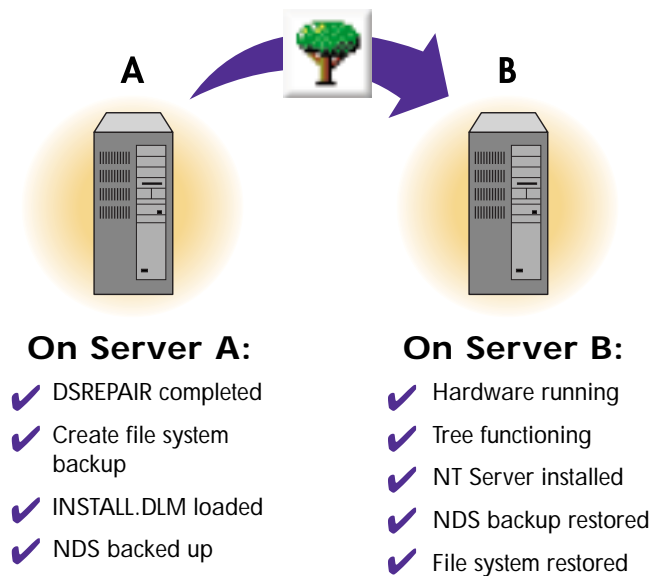To prepare for the upgrade, you will need to make a backup of the Directory Information Base (dib). Because other servers in the eDirectory tree expect the server to come back online quickly, you should complete the upgrade promptly and restore eDirectory information on the server as soon as possible.

Preparing for a Hardware Change on Linux or Solaris

Use the following checklist to determine if you are ready to start this procedure:

❑ Ensure that the tree on Server A is healthy by running the ndsrepair utility on the server that holds the master of the Tree partition. Also ensure that the time is synchronized.

❑ Run the ndsrepair utility on the database of Server A. Ensure that the tree is in good health and the server A is synchronized completely.

❑ Make a backup of the server dib. Complete the following steps to make a backup of all eDirectory related files:

  ◆ Stop the eDirectory daemon on Server A.

  ◆ Make a backup of the /var/nds directory.

  ◆ Make a backup of /etc/nds.conf, /etc/nsswitch.conf, and /etc/pam.conf.

❑ Bring up Server B, and copy the above files to an appropriate location.

❑ Start the eDirectory daemon and run the ndstrace utility to ensure that the server is communicating with other eDirectory servers in the replica ring.

❑ Run the ndswskey utility if Single Sign-on (SSO) was installed.

Creating a Backup of eDirectory on Linux or Solaris

Although there is no utility available to back up all eDirectory-related data on Linux or Solaris systems, you can use the ndsbackup utility to make a backup of the database. For more information, see "Using Backup and Restore

Before upgrading hardware or changing system configuration, tar the following files and directories.

- The /var/nds directory

- The /etc/nds.conf file

- The /etc/nsswitch.conf and /etc/pam.conf files, if the Account Management component is installed

Restoring eDirectory Information after a Hardware Upgrade on Linux or Solaris

If Server B does not function properly, and you need to make Server A functional immediately, complete the following:

1 Unplug Server B's network cable or down the server.

2 Re-attach Server A to the network and start it.

3 Remove eDirectory from Server B and try the upgrade again.

Restoring eDirectory on NetWare after a Hardware Failure

Instructions in the following sections explain how to restore eDirectory for a *specific server* when you have a hardware failure. For information on backing up and restoring eDirectory on NetWare on an *entire tree*, refer to "Using Backup and Restore Services on NetWare" on page 423.

In a multiple-server environment, one server can go down while the rest of the servers in its replica list remain intact.

If the hard disk containing volume SYS: on one server becomes damaged, the entire server is affected. A hard disk failure involving volume SYS: affects the entire server and halts all NetWare operating system activities. Because the eDirectory files are stored on volume SYS:, losing this volume is equivalent to removing NetWare and eDirectory from the file server.

To restore eDirectory to a single server, you will need to complete the following procedures:

❑ "Retrieving Server-Specific Information Files" on page 462

❑ "Cleaning Up the Replica Ring" on page 463

❑ "Installing the New Server" on page 464

❑ "Adding the New Server to Previously Defined Replica Rings" on page 466

❑ "Verifying Successful eDirectory Restore" on page 467

Step-by-step instructions are contained in the following sections.

Retrieving Server-Specific Information Files

The following procedure assumes you have a current backup of the server-specific information for the failed server.

IMPORTANT: If you do not have a backup for the failed server, you will need to remove the server object from the tree and restore eDirectory as explained in Novell Technical Information Document #10011467 (http://support.novell.com/cgi-bin/search/searchtid.cgi?/10011467.htm).

1 From your backup host server, run SMS™ and restore server-specific information files from a tape backup.

The server-specific information files are restored to a subdirectory in SYS:\SYSTEM on the server you have selected. The subdirectory name is a DOS 8.3 name derived from the source server name.

The following files are restored:

SERVDATA.NDS contains server-specific NDS information that allows trustee assignments and other NetWare information to be preserved.

DSMISC.LOG contains a list of replicas, their types, and other servers that participated in the failed server's replica ring.

VOLSINFO.TXT contains information about the server's volumes, including name space, compression, and data migration information.

STARTUP.NCF is the NetWare server boot file that loads the server's disk driver and name spaces and some SET parameters.

AUTOEXEC.NCF is the NetWare server batch file that loads modules and sets the OS configuration.

2 Make these files easily accessible for use during the remaining process.

For complete information on using SMS, see the Storage Management Services Administration Guide.

Cleaning Up the Replica Ring

This procedure explains how to

- **Reassign master replicas:** If the failed server holds a master replica of any partition, you will use DSREPAIR to designate a new master replica on a different server in the replica list.

- **Remove replica list references to the failed server:** Each server participating in replica rings that included the failed server must be told that the failed server is no longer available.

1 Use the information contained in the DSMISC.LOG file to determine which replicas were stored on the failed server.

2 At the console of one of the servers that shared a replica with the failed server, load **DSREPAIR -a**.

WARNING: If used incorrectly, using DSREPAIR with the -a option can cause damage to your tree. For more information on these options, refer to Novell Technical Information Document #2938493 (http://support.novell.com/cgi-bin/search/searchtid.cgi?/2938493.htm).

3 Select Advanced Options Menu > Replica and Partition Operations.

Use this option in DSREPAIR only when the master replica of a partition is lost because of a server or hardware failure.

4 Select the partition you want to edit.

5 Select View Replica Ring to see a list of servers that have replicas of the partition.

6 Select the server you want to hold the master replica > select Designate This Server as the New Master Replica.

The replica ring now has a new Master Replica. All replicas participating in the ring will be told that there is a new master.

7 Step back one menu to where you can select View Replica Ring > choose the name of the failed server.

8 Select Remove This Server from the Replica Ring > log in as Admin.

9 After reading the explanation message, enter your agreement to continue.

10 Exit DSREPAIR.

All servers participating in that replica ring will be notified.

11 Repeat this procedure as applicable on one server for each replica ring that the failed server participated in.

Installing the New Server

This installation procedure accomplishes the following:

- Re-installs NetWare
- Restores eDirectory
- Restores the file system

1 Install the new hard disk or server hardware.

Follow any instructions provided by the manufacturer to verify that the server's hard disks are working. The new hard disk should have the same (or larger) storage capacity as the drive it replaces. Use the VOLSINFO.TXT file to verify configuration information.

2 Install NetWare.

See the NetWare Installation manual for details about starting install.

3 During installation, re-enter the same server name, eDirectory context, and network address that the server had prior to the failure.

Use the STARTUP.NCF and AUTOEXEC.NCF files for needed information.

4 When prompted to enter a tree name, enter a *new* tree name so the server will install into its own temporary tree.

Using a new tree name prevents server identity conflicts.

5 Copy the restored server-specific files from the backup host server to the newly installed server.

These files need to be available for reference for the following steps.

6 At the server console prompt, type NWCONFIG.

7 Select Directory Services Options > select Remove Directory Services from This Server > log in to the tree.

Removing the directory allows you to restore the failed server's identity completely and restore it to the original tree.

8 Select Directory Services Options > Directory Backup and Restore Options > Restore Local Server Information after a Hardware Failure.

9 Press F3 to specify the path to SERVDATA.NDS.

eDirectory is restored at this time, using the information contained in SERVDATA.NDS. Once this is done, eDirectory is fully functional on the server, except that the partitions and replicas have not yet been re-established. Reestablishing replica rings is done after install.

10 Prepare the restored directory for integration into the original tree by executing the following DSREPAIR commands in the order specified:

WARNING: The following DSREPAIR switch commands should only be used in this procedure and should be executed in the order given. DO NOT use these commands for general DSREPAIR operations.

`DSREPAIR -si` (Repairs replica numbers on partition objects)

`DSREPAIR -rd` (Local database repair)

`DSREPAIR -rn` (Repairs network addresses)

`DSREPAIR -sr` (Requests local schema switch)

This is a clean up process. It's ok to see error messages displayed during the clean up.

11 Use DSTRACE to verify that the schema has synchronized fully. From the NetWare console, type

SET DSTRACE = ON (Activates the NDS transactions screen)

SET DSTRACE = +SCHEMA (Displays schema information)

SET DSTRACE = IN (Monitors inbound synchronization traffic)

Press Ctrl+Esc > select Directory Services to view the trace screen. Watch for the message "All processed = YES."

Once the schema is synchronized, you are ready to do a file system restore.

12 From the server console enter **TSA500.nlm** to load the file system TSA.

Use SMS to start the restoration of the file system for each volume affected by the failure.

If the server that failed was the host server for the backup program, first take the steps necessary to reinstall the backup software and storage device drivers.

13 After the restoration of the file system is completed, down the server and bring it back up.

Adding the New Server to Previously Defined Replica Rings

This process ensures replication on the new server will be at the same level as it was on the failed server.

Use DSMISC.LOG to aid in this process. It contains a copy of the replica list that resided on the server at the time the backup was made.

1 From the server console, enter **edit dsmisc.log** to view the log file contents.

2 Re-establish replicas on the failed server using ConsoleOne.

2a In the left pane, right-click the root container of the partition that you want to replicate > click Views > click Partition and Replica View.

2b On the toolbar, click Add Replica.

2c Next to the Server Name field, click the browse button > select the server you just restored > click OK.

2d Select the type of replica you want > click OK.

If the failed server had master replicas, you can designate that same set of replicas as master replicas.

For information on using ConsoleOne, see the online help.

Verifying Successful eDirectory Restore

Use ConsoleOne to verify a successful restoration.

❑ Check to see that the server's replica states are On.

Right click the server in the left pane > click Views >Partition and Replica View.

❑ Confirm that trustee assignments for the file system are as you expect.

You can use the NDS User Security Reports to get a complete listing of trustees and their assigned rights. See Generating Reports in *ConsoleOne User Guide*.

Restoring eDirectory on NT after a Hardware Failure

Instructions in the following section give information you need to know about eDirectory on a *specific server* when you have a hardware failure. For information on backing up and restoring an *entire* eDirectory tree on Windows NT, refer to "Using Backup and Restore Services on Windows NT" on page 424.

In a multiple-server environment, one server can go down while the rest of the servers in its replica list remain intact.

If the hard disk containing the eDirectory files on one NT server becomes damaged, it is equivalent to removing eDirectory from the server.

To restore eDirectory to a single server, you will need to complete the following procedures:

❑ "Retrieving Server-Specific Information Files" on page 468

❑ "Cleaning Up the Replica Ring" on page 468

❑ "Installing the New Server" on page 469

❑ "Adding the New Server to Previously Defined Replica Rings" on page 471

❑ "Verifying Successful eDirectory Restore" on page 471

Step-by-step instructions are contained in the following sections.

Retrieving Server-Specific Information Files

The following procedure assumes you have a current backup of the server-specific information for the failed server.

IMPORTANT: If you do not have a backup for the failed server, you will need to remove the server object from the tree and restore eDirectory as explained in Novell Technical Information Document #10011467 (http://support.novell.com/cgi-bin/search/searchtid.cgi?/10011467.htm).

1 From your backup host server, run SMS and restore local server information files from a tape backup.

The local server information files should be created and backed up periodically as part of your disaster recovery plan. These files are created in the novell\nds\DIBfiles directory when you load INSTALL.DLM from the NDS Services console and select the Backup Local Server Information option.

The following files should be restored:

$SVNDS.BAK contains server-specific NDS information that allows trustee assignments and other NetWare information to be preserved.

DSBACKUP.LOG contains a list of replicas, their types, and other servers that participated in the failed server's replica ring.

2 Make these files easily accessible for use during the remaining process.

For complete information on using SMS, see the Storage Management Services Administration Guide.

Cleaning Up the Replica Ring

This procedure explains how to

 * **Reassign master replicas:** If the failed server holds a master replica of any partition, you will use DSREPAIR to designate a new master replica on a different server in the replica list.

 * **Remove replica list references to the failed server:** Each server participating in replica rings that included the failed server must be told that the failed server is no longer available.

1 Use the information contained in the DSBACKUP.LOG file to determine which replicas were stored on the failed server.

2 At the NDS Services console, select DSREPAIR.DLM and enter **-a** in the Command Line field, then choose Start.

WARNING: If used incorrectly, using DSREPAIR with the -a option can cause damage to your tree. For more information on these options, refer to the Technical Support Web site Solution 2938493 (http://support.novell.com/cgi-bin/search/searchtid.cgi?/2938493.htm).

3 In the DSREPAIR browser window, expand the Partitions list > select the replica you want to designate as the master > click Partitions > click Designate This Server as the New Master Replica.

The replica ring now has a new Master Replica. All replicas participating in the ring will be told that there is a new master.

4 In the DSREPAIR browser window, expand the Partitions list > select the replica from which you want to remove the server > select the failed server > click Partitions > click Replica Ring > click Remove Server from Ring.

5 Exit DSREPAIR.

All servers participating in that replica ring will be notified of replica ring updates.

6 Repeat this procedure as applicable on one server for each replica ring that the failed server participated in.

7 Force synchronization of all replicas in DSRepair. Choose Replica Rings > Send all objects to every replica in the ring.

Installing the New Server

This procedure covers installing the Windows Operating System and installing eDirectory.

1 Install the new hard disk or server hardware.

Follow any instructions provided by the manufacturer to verify that the server's hard disks are working. The new hard disk should have the same (or larger) storage capacity as the drive it replaces. Use the local server information files to verify configuration information.

2 Install the Windows 2000 or NT operating system.

3 Install eDirectory on NT on the new server. See .

4 When prompted to enter a tree name, enter a *new* tree name so the server will install into its own temporary tree.

5 From the NDS Services console, select INSTALL.DLM > click Start > click Remove Directory Services > click Finish.

6 Enter the administrator name and password > click OK and follow the prompts.

eDirectory is removed.

7 From the NDS Services console select INSTALL.DLM > click Restore Local Server Information after a Hardware Failure > click Next.

8 Specify the path to the $SVNDS.BAK file > click OK.

eDirectory is restored at this time using the information in $SVNDS.BAK, but you still need to prepare the directory for integration and re-establish the partitions and replicas.

9 Prepare the restored directory for integration into the original tree by executing the following DSREPAIR commands from the newly restored server in the order specified:

WARNING: The following DSREPAIR commands should only be used in this procedure and should be executed in the order given. DO NOT use these commands for general DSREPAIR operations.

9a `DSREPAIR -si` (Repairs replica numbers on partition objects)

9b Choose Repair > Local database repair

9c Choose Servers > Repair all network addresses

9d Choose Schema >Reset local schema

This is a clean up process. It's ok to see error messages displayed during the clean up.

10 Use DSTRACE to verify that the schema has synchronized fully. From the NDS Services console, select dstrace > click Start. Choose Edit > Options and mark Schema and sync incoming, then choose OK.

Watch for the message "All processed = YES."

Once the schema is synchronized, you are ready to re-establish replicas.

Adding the New Server to Previously Defined Replica Rings

This process ensures replication on the new server will be at the same level as it was on the failed server.

1 View the contents of DSBACKUP.LOG.

It contains a copy of the replica list that resided on the server at the time the backup was made.

2 Re-establish replicas on the failed server using ConsoleOne.

2a In the left pane, right-click the root container of the partition that you want to replicate > click Views > click Partition and Replica View.

2b On the toolbar, click Add Replica.

2c Next to the Server Name field, click the browse button > select the server you just restored > click OK.

2d Select the type of replica you want > click OK.

If the failed server had master replicas, you can designate that same set of replicas as master replicas.

For information on using ConsoleOne, see the online help.

Verifying Successful eDirectory Restore

Use ConsoleOne to verify that the server's replica states are On

Right-click the server in the left pane > click Views > Partition and Replica View.

14 Troubleshooting Novell eDirectory

The following sections give suggestions and resources for solving issues with Novell® eDirectory™:

- "Resolving Error Codes" on page 473
- "Troubleshooting eDirectory on Windows NT/2000" on page 473
- "Troubleshooting LDIF Files" on page 476
- "Troubleshooting eDirectory on Linux and Solaris" on page 497
- "Troubleshooting Dynamic Groups" on page 524
- "Troubleshooting ConsoleOne" on page 525

Resolving Error Codes

For a complete list of eDirectory error codes, see the Novell Error Codes Web page (http://www.novell.com/documentation/lg/nwec/index.html).

Naming Objects

When you use special characters while naming objects, the "No such parent" error message appears. Avoid using any of the following special characters while naming objects:

\ /, * ? .

Troubleshooting eDirectory on Windows NT/2000

This section includes information for troubleshooting eDirectory on Windows* NT* and Windows 2000 networks.

Recovering from eDirectory Server Problems

Recovering from eDirectory Replica Problems

eDirectory offers the Novell robust directory service and the fault tolerance inherent in replication. Replication allows you to keep copies of the eDirectory database, or portions of it, on multiple servers at once.

You should always keep multiple replicas of eDirectory partitions. If you do so and one replica becomes corrupted or is lost because of a failed hard disk, you can delete that replica using ConsoleOne™ and replace it with a new one from the intact replica.

The eDirectory Server (on the NT/2000 Server) Won't Start

If the eDirectory server fails to start when you boot the NT/2000 server, a message will notify you that the service failed to start.

If there are no other eDirectory database replicas, users can't log in.

If there are other replicas, logging in might be slow and you will see communication errors and synchronization errors on the servers holding those replicas.

- The eDirectory server entries in the Windows Registry might have been edited, or the Windows Registry might be corrupt.

- eDirectory database files might have been corrupted or deleted.

- If the eDirectory server can't start because another service didn't start, you can get more information from Start > Programs > Administrative Tools > Event Viewer.

  You'll need to resolve the related-service problem before starting the eDirectory server.

- The Registry or eDirectory executable files are corrupted or lost. Run the SAMMIG.EXE utility in the system directory. Select Uninstall NDS® on Windows NT and include new eDirectory information in the NT domain. Continue with the Uninstall until completed. Then restart SAMMIG.EXE and proceed to install eDirectory.

- Database files have been corrupted or deleted. If the eDirectory server comes up on the NT server but the service can't open the eDirectory database files, see

The NT/2000 Server Can't Open the eDirectory Database Files

If the eDirectory server can't open the database files, a message on the NT/2000 server will notify you.

If there are no other database replicas, users can't log in.

If there are other replicas, logging in might be slow and you will see communication errors and synchronization errors on the servers holding those replicas.

- ◆ The database files might have been corrupted through disk errors on the NT/2000 server.

- ◆ Someone might have deleted one or more of the database files.

If other replicas of the eDirectory database exist, complete the following steps:

1 Start ConsoleOne from an administrative workstation.

2 Select the corrupted replica and remove it from the replica ring.

3 Run the SAMMIG.EXE utility in the system directory (usually C:\WINNT\SYSTEM32) on the NT server or from the Start menu: Start > Programs > Administrative Tools (Common) > Migration Tool for NetWare®.

4 Select the option to create a new replica on the eDirectory server.

If this eDirectory server holds the only replica of the partition, complete the following steps:

1 Run the SAMMIG.EXE utility in the system directory (usually C:\WINNT\SYSTEM32) on the NT Server or from the Start menu: Start > Programs > Administrative Tools (Common) > Migration Tool for NetWare.

2 Select Uninstall NDS on Windows NT/2000 and revert to the previous NT/2000 domain state.

3 Continue with the Uninstall until it has completed.

4 Restart the Migration Tool for NetWare and proceed to install eDirectory on Windows NT/2000.

5 Move the User objects from the NT/2000 domain to the eDirectory tree.

Restoring eDirectory on Windows NT/2000 after an Emergency Repair

When you are forced to do an emergency repair on an NT/2000 server, and there is no Emergency Repair disk, or the Emergency Repair disk was created

before an eDirectory installation, the eDirectory client is removed and Registry settings are deleted. The NDS4NTER.EXE utility both restores the necessary Registry settings and reloads eDirectory files.

Run NDS4NTER.EXE from the \i386\GOODIES directory.

After an emergency repair is performed, run the Emergency Repair utility from the CD. The utility will first restore some of the Registry settings, then it will launch the eDirectory installation. The installation will copy the files and then you must select the reboot option. After rebooting, users will have access to the migrated domains.

Log Files

MODSCHEMA.LOG

The MODSCHEMA.LOG file contains the results of all schema extensions that are applied when an eDirectory server is installed into an existing tree. Each line of the log states which class/attribute is being added/modified and gives the status of the modification attempt.

This log is created or overwritten each time the install process is run, so it only represents the results of the last attempt. In addition to the eDirectory schema extensions, this log contains the results of any other schema extensions (such as LDAP or SAS) applied by the DSINSTALL front end prior to adding the new eDirectory server.

This log will not be generated when a standalone server is installed or if the version of the target server is NDS 7.01 or later.

DSINSTALL.LOG

The first part of the log lists environment variables that are set. The second part contains status messages documenting the eDirectory installation process.

Troubleshooting LDIF Files

The Novell Import Conversion Export utility lets you easily import LDIF files into and export LDIF files from eDirectory. For more information, see

In order for an LDIF import to work properly, you must start with an LDIF file that the Novell Import Conversion Export utility can read and process. This

section describes the LDIF file format and syntax and provides examples of correct LDIF files.

Understanding LDIF

LDIF is a widely used file format that describes directory information or modification operations that can be performed on a directory. LDIF is completely independent of the storage format used within any specific directory implementation, and is typically used to export directory information from and import data to LDAP servers.

LDIF is generally easy to generate. This makes it possible to use tools like awk or perl to move data from a proprietary format into an LDAP directory. You can also write scripts to generate test data in LDIF format.

LDIF File Format

Novell Import Conversion Export imports require LDIF 1 formatted files. The following are the basic rules for an LDIF 1 file:

* The first non-comment line must be version: 1.

* A series of one or more records follows the version.

* Each record is composed of fields, one field per line.

* Lines are separated by either a new line or a carriage return/new line pair.

* Records are separated by one or more blank lines.

* There are two distinct types of LDIF records: content records and change records. An LDIF file can contain an unlimited number of records, but they all must be of the same type. You can't mix content records and change records in the same LDIF file.

* Any line beginning with the pound sign (#) is a comment and is ignored when processing the LDIF file.

LDIF Content Records

An LDIF content record represents the contents of an entire entry. The following is an example of an LDIF file with four content records:

```
1  version: 1
2  dn: c=US
3  objectClass: top
4  objectClass: country
```

```
 5
 6  dn: l=San Francisco, c=US
 7  objectClass: top
 8  objectClass: locality
 9  st: San Francisco
10
11 dn: ou=Artists, l=San Francisco, c=US
12 objectClass: top
13 objectClass: organizationalUnit
14 telephoneNumber: +1 415 555 0000
15
16 dn: cn=Peter Michaels, ou=Artists, l=San Francisco, c=US
17 sn: Michaels
18 givenname: Peter
19 objectClass: top
20 objectClass: person
21 objectClass: organizationalPerson
22 objectClass: iNetOrgPerson
23 telephonenumber: +1 415 555 0001
24 mail: Peter.Michaels@aaa.com
25 userpassword: Peter123
26
```

This LDIF file is composed of the following parts:

Table 140   **LDIF File Components**

Part	Description
Version Specifier	The first line of an LDIF file contains the version. Zero or more spaces are allowed between the colon and the version number, which is currently defined to be 1.
	If the version line is missing, any application processing the LDIF file is allowed to assume that the file is version 0. It's also possible that the LDIF file could be rejected as syntactically incorrect. Novell utilities that process LDIF assume a file version of 0 when the version line is missing.

Part	Description
Distinguished Name Specifier	The first line of every content record (lines 2, 6, 11, and 16 in the example above) specifies the DN of the entry that it represents.
	The DN specifier must take one of the following two forms:
	◆ dn: *safe_UTF-8_distinguished_name*
	◆ dn:: *Base64_encoded_distinguished_name*
Line Delimiters	The line separator can be either a line feed or a carriage return/line feed pair. This resolves a common incompatibility between Linux* and Solaris* text files, which use a line feed as the line separator, and MS-DOS and Windows text files, which use a carriage return/line feed pair as the line separator.
Record Delimiters	Blank lines (lines 5, 10, 15, and 26 in the example above) are used as record delimiters.
	Every record in an LDIF file including the last record must be terminated with a record delimiter (one or more blank lines). Although some implementations will silently accept an LDIF file without a terminating record delimiter, the LDIF specification requires it.
Attribute Value Specifier	All other lines in a content records are value specifiers. Value specifiers must take on one of the following three forms:
	◆ Attribute description: *value*
	◆ Attribute description:: *Base64_encoded_value*
	◆ Attribute description:< *URL*

LDIF Change Records

LDIF change records contain modifications to be made to a directory. Any of the LDAP update operations (add, delete, modify, and modify DN) can be represented in an LDIF change record.

LDIF change records use the same format for the distinguished name specifier, attribute value specifier, and record delimiter as LDIF content

records. (See "LDIF Content Records" on page 477 for more information.) The presence of a changetype field is what distinguishes an LDIF change record from an LDIF content record. A changetype field identifies the operation specified by the change record.

A changetype field can take one of the following five forms:

Changetype Field Forms

Changetype	Description
changetype: add	A keyword indicating that the change record specifies an LDAP add operation.
changetype: delete	A keyword indicating that the change record specifies an LDAP delete operation.
changetype: moddn	A keyword indicating that the change record specifies an LDAP modify DN operation if the LDIF processor is bound to the LDAP server as a version 3 client or a modify RDN operation if the LDIF processor is bound to the LDAP server as a version 2 client.
changetype: modrdn	A synonym for the moddn change type.
changetype: modify	A keyword indicating that the change record specifies an LDAP modify operation.

The Add Change Type

An add change record looks just like a content change record (see "LDIF Content Records" on page 477) with the addition of the changetype: add field immediately before any attribute value fields.

All records must be the same type. You can't mix content records and change records.

```
 1 version: 1
 2 dn: c=US
 3 changetype: add
 4 objectClass: top
 5 objectClass: country
 6
 7 dn: l=San Francisco, c=US
 8 changetype: add
 9 objectClass: top
10 objectClass: locality
11 st: San Francisco
```

```
12
14 dn: ou=Artists, l=San Francisco, c=US
15 changetype: add
16 objectClass: top
17 objectClass: organizationalUnit
18 telephoneNumber: +1 415 555 0000
19
20 dn: cn=Peter Michaels, ou=Artists, l=San Francisco, c=US
21 changetype: add
22 sn: Michaels
23 givenname: Peter
24 objectClass: top
25 objectClass: person
26 objectClass: organizationalPerson
27 objectClass: iNetOrgPerson
28 telephonenumber: +1 415 555 0001
29 mail: Peter.Michaels@aaa.com
30 userpassword: Peter123
31
```

The Delete Change Type

Because a delete change record specifies the deletion of an entry, the only fields required for a delete change record are the distinguished name specifier and a delete change type.

The following is an example of an LDIF file used to delete the four entries created by the LDIF file shown in "The Add Change Type" on page 480.

IMPORTANT: To delete entries you have previously added, reverse the order of the entries. If you don't do this, the delete operation will fail since the container entries would not be empty.

```
 1 version: 1
 2 dn: cn=Peter Michaels, ou=Artists, l=San Francisco, c=US
 3 changetype: delete
 4
 5 dn: ou=Artists, l=San Francisco, c=US
 8 changetype: delete
 9
10 dn: l=San Francisco, c=US
11 changetype: delete
12
13 dn: c=US
14 changetype: delete
15
```

The Modify Change Type

The modify change type lets you to specify the addition, deletion, and replacement of attribute values for an entry that already exists. Modifications take one of the following three forms:

Table 142   Modification Specifier Elements

Modification Specifier Elements	Description
add: attribute type	A keyword indicating that subsequent attribute value specifiers for the attribute type should be added to the entry.
delete: attribute type	A keyword indicating that values of the attribute type are to be deleted. If attribute value specifiers follow the delete field, the values given are deleted.
	If no attribute value specifiers follow the delete field, then all values are deleted. If the attribute has no values, this operation will fail, but the desired effect will still be achieved because the attribute had no values to be deleted.
replace: attribute type	A keyword indicating that the values of the attribute type are to be replaced. Any attribute value specifiers that follow the replace field become the new values for the attribute type.
	If no attribute value specifiers follow the replace field, the current set of values is replaced with an empty set of values (which causes the attribute to be removed). Unlike the delete modification specifier, if the attribute has no values, the replace will still succeed. The net effect in both cases is the same.

The following is an example of a modify changetype that will add an additional telephone number to the cn=Peter Michaels entry.

```
1 version: 1
2 dn: cn=Peter Michaels, ou=Artists, l=San Francisco, c=US
3 changetype: modify
```

```
 4 # add the telephone number to cn=Peter Michaels
 4 add: telephonenumber
 5 telephonenumber: +1 415 555 0002
 6
```

Just as you can combine a mixture of modifications in a single LDAP modify request, you can specify multiple modifications in a single LDIF record. A line containing only the hyphen (-) character is used to mark the end of the attribute value specifications for each modification specifier.

The following example LDIF file contains a mixture of modifications:

```
 1 version: 1
 2
 3 # An empty line to demonstrate that one or more
 4 # line separators between the version identifier
 5 # and the first record is legal.
 6
 7 dn: cn=Peter Michaels, ou=Artists, l=San Francisco, c=US
 8 changetype: modify
 9 # Add an additional telephone number value.
10 add: telephonenumber
11 telephonenumber: +1 415 555 0002
12 -
13 # Delete the entire fascimiletelephonenumber attribute.
14 delete: facsimileTelephoneNumber
15 -
16 # Replace the existing description (if any exists)
17 # with two new values.
18 replace: description
19 description: guitar player
20 description: solo performer
21 -
22 # Delete a specific value from the telephonenumber
23 # attribute.
24 delete: telephonenumber
25 telephonenumber: +1 415 555 0001
26 -
27 # Replace the existing title attribute with an empty
28 # set of values, thereby causing the title attribute to
29 # be removed.
30 replace: title
31 -
32
```

The Modify DN Change Type

The modify DN change type lets you rename an entry, move it, or both. This change type is composed of two required fields and one optional field.

Table 143   **Modify DN Change Type Fields**

Field	Description
newrdn (required)	Gives the new name for the entry that will be assigned while processing this record. The new RDN specifier must take of the following two forms:
	◆ newrdn: *safe_UTF-8_relative_distinguished_name*
	◆ newrdn:: *Base64_encoded_relative_distinguished_name*
	The new RDN specifier is required in all LDIF records with a modify DN change type.
deleteoldrdn (required)	The delete old RDN specifier is a flag that indicates whether the old RDN should be replaced by the newrdn or if it should be kept. It takes one of the two following forms:
	◆ deleteoldrdn: 0
	Indicates that the old RDN value should be kept in the entry after it is renamed.
	◆ deleteoldrdn: 1
	Indicates that the old RDN value should be deleted when the entry is renamed.

Field	Description
newsuperior (optional)	The new superior specifier gives the name of the new parent that will be assigned to the entry while processing the modify DN record. The new superior specifier must take of the following two forms: • newsuperior: *safe_UTF-8_distinguished_name* • newsuperior:: *Base64_encoded_distinguished_name* The new superior specifier is optional in LDIF records with a modify DN change type. It is only given in cases where you want to reparent the entry.

The following is an example of a modify DN changetype that shows how to rename an entry:

```
1 version: 1
2
3 # Rename ou=Artists to ou=West Coast Artists, and leave
4 # its old RDN value.
5 dn: ou=Artists,l=San Francisco,c=US
6 changetype: moddn
7 newrdn: ou=West Coast Artists
8 deleteoldrdn: 1
9
```

The following is an example of a modify DN changetype that shows how to move an entry:

```
1 version: 1
2
3 # Move cn=Peter Michaels from
4 # ou=Artists,l=San Francisco,c=US to
5 # ou=Promotion,l=New York,c=US and delete the old RDN.
5 dn: cn=Peter Michaels,ou=Artists,l=San Francisco,c=US
6 changetype: moddn
7 newrdn: cn=Peter Michaels
8 deleteoldrdn: 1
9 newsuperior: ou=Promotion,l=New York,c=US
10
```

The following is an example of a modify DN changetype that shows how to move an entry and rename it at the same time:

```
1 version: 1
2
3 # Move ou=Promotion from l=New York,c=US to
4 # l=San Francisco,c=US and rename it to
5 # ou=National Promotion.
5 dn: ou=Promotion,l=New York,c=US
6 changetype: moddn
7 newrdn: ou=National Promotion
8 deleteoldrdn: 1
9 newsuperior: l=San Francisco,c=US
10
```

IMPORTANT: The LDAP 2 modify RDN operation doesn't support moving entries. If you try to move an entry using the LDIF newsuperior syntax with an LDAP 2 client, the request will fail.

Line Folding within LDIF Files

To fold a line in an LDIF file, simply insert a line separator (a newline or a carriage return/newline pair) followed by a space at the place where you want the line folded. When the LDIF parser encounters a space at a beginning of the line, it knows to concatenate the rest of the data on the line with the data on the previous line. The leading space is then discarded.

You should not fold lines in the middle of a multi-byte UTF-8 character.

The following is an example of an LDIF file with a folded line (see lines 13 and 14):

```
 1 version: 1
 2 dn: cn=Peter Michaels, ou=Artists, l=San Francisco, c=US
 3 sn: Michaels
 4 givenname: Peter
 5 objectClass: top
 6 objectClass: person
 7 objectClass: organizationalPerson
 8 objectClass: iNetOrgPerson
 9 telephonenumber: +1 415 555 0001
10 mail: Peter.Michaels@aaa.com
11 userpassword: Peter123
12 description: Peter is one of the most popular music
13  ians recording on our label. He's a big concert dr
14  aw, and his fans adore him.
15
```

Debugging LDIF Files

If you have problems with an LDIF file, consider the following:

- ◆ "Enabling Forward References" on page 488
- ◆ "Checking the Syntax of LDIF Files" on page 489
- ◆ "Using the LDIF Error File" on page 490
- ◆ "Using LDAP SDK Debugging Flags" on page 491

Enabling Forward References

You might occasionally encounter LDIF files in which a record to add one entry comes before a record to add its parents. When this happens, an error is generated because the new entry's parent does not exist when the LDAP server attempts to add the entry.

To solve this problem, simply enable the use of forward references. When you enable the creation of forward references and an entry is going to be created before its parent exists, a placeholder called a forward reference is created for the entry's parent to allow the entry to be successfully created. If a later operation creates the parent, the forward reference is changed into a normal entry.

It is possible that one or more forward references will remain once your LDIF import is complete (if, for example, the LDIF file never created the parent for an entry). In this case, the forward reference will appear as an Unknown object in ConsoleOne. Although you can search on a forward reference entry, you cannot read attributes (except objectClass) from the forward reference entry because it does not have any attributes or attribute values. However, all LDAP operations will work normally on the real object entries located below the forward reference.

Identifying Forward Reference Entries

Forward reference entries have an object class of Unknown and also have their internal NDS EF_REFERENCE entry flag set. In ConsoleOne, entries with an object class of Unknown are represented by a round yellow icon with a question mark in the center. You can use LDAP to search for objects with an Unknown object class, although there is currently no way to access the entry flag settings through LDAP to be sure that they are forward reference entries.

Changing Forward Reference Entries into Normal Objects

You can change a forward reference entry into an normal object by simply creating it (using, for example, an LDIF file or an LDAP client request). When you ask eDirectory to create an entry that already exists as a forward reference, eDirectory transforms the existing forward reference entry into the object you asked it to create.

Using the Novell eDirectory Import/Export Wizard

To enable forward references during an LDIF import:

1 In ConsoleOne, select Wizard > NDS Import/Export.

2 Click Import LDIF File > Next.

3 Enter the name of the LDIF file containing the data you want to import >
click Next.

4 Select the LDAP server where the data will be imported.

5 Click Advanced > Allow Forward References > Close.

6 Click Next > Finish to begin the LDIF import.

To enable forward references during a data to data server migration:

1 In ConsoleOne, select Wizard > NDS Import/Export.

2 Click Migrate Data between LDAP Servers > Next.

3 Select the LDAP server holding the entries you want to migrate > click
Next.

4 Specify search criteria for the entries you want to migrate > click Next.

5 Select the LDAP server where the data will be migrated.

6 Click Advanced > Allow Forward References > Close.

7 Click Next > Finish.

**Using the Novell Import Conversion Export Utility Command Line
Interface**

To enable forward references in the command line interface, use the -F LDAP
destination handler option.

For more information, see "LDIF Destination Handler Options" on page 184.

Checking the Syntax of LDIF Files

You can check the syntax of an LDIF file before you process the records in the
file by using the Display Operations but Do Not Perform LDIF source handler
option.

The LDIF source handler always checks the syntax of the records in an LDIF
file as it processes them. Using this option disables the processing of the
records and lets you verify the syntax.

Using the Novell eDirectory Import/Export Wizard

To check your syntax during an LDIF import:

1 In ConsoleOne, select Wizard > NDS Import/Export.

2 Click Import LDIF File > Next.

3 Enter the name of the LDIF file containing the data you want to import > click Advanced.

4 Click Display Operations but Do Not Perform > Close > Next.

5 Select the LDAP server where the data will be imported.

6 Click Next > Finish to begin the LDIF import.

Using the Novell Import Conversion Export Utility Command Line Interface

To check the syntax of an LDIF file in the command line interface, use the -n LDIF source handler option.

For more information, see "LDIF Source Handler Options" on page 183.

Using the LDIF Error File

The Novell Import Conversion Export utility automatically creates an LDIF file listing any records that failed processing by the destination handler. You can edit the LDIF error file generated by the utility, fix the errors, then re-apply it to the server to finish an import or data migration that contained failed records.

Using the Novell eDirectory Import/Export Wizard

1 In ConsoleOne, select Wizard > NDS Import/Export.

2 Click the task you want to perform.

3 Click Advanced.

4 In the Log File field, specify a filename where output messages (including error messages) will be logged.

5 In the LDIF Output File for Failed Records field, specify a filename where entries that fail are output in LDIF format.

You can use this file to examine or correct errors.You can also reapply a modified (corrected) version of this file to the directory.

6 Click Close.

7 Follow the online instructions to finish your selected task.

Using the Novell Import Conversion Export Utility Command Line Interface

To configure error log options in the command line utility, use the -l general option.

For more information, see "General Options" on page 181.

Using LDAP SDK Debugging Flags

To understand some LDIF problems, you might need to see how the LDAP client SDK is functioning. You can set the following debugging flags for the LDAP source handler, the LDAP destination handler, or both.

Table 144   LDAP SDK Debugging Flags

Value	Description
0x0001	Trace LDAP function calls.
0x0002	Print information about packets.
0x0004	Print information about arguments.
0x0008	Print connections information.
0x0010	Print BER encoding and decoding information.
0x0020	Print search filter information.
0x0040	Print configuration information.
0x0080	Print ACL information.
0x0100	Print statistical information.
0x0200	Print additional statistical information.
0x0400	Print shell information.
0x0800	Print parsing information.
0xFFFF (-1 Decimal)	Enable all debugging options.

To enable this functionality, use the -e option for the LDAP source and LDAP destination handlers. The integer value you give for the -e option is a bitmask that enables various types of debugging information in the LDAP SDK.

For more information, see "LDAP Source Handler Options" on page 184 and "LDAP Destination Handler Options" on page 188.

Using LDIF to Extend the Schema

Because LDIF can represent LDAP update operations, you can use LDIF to modify the schema.

Adding a New Object Class

To add a class, simply add an attribute value that conforms to the specification for NDSObjectClassDescription to the objectClasses attribute of the subschemaSubentry.

```
NDSObjectClassDescription = "(" whsp
   numericoid whsp
   [ "NAME" qdescrs ]
   [ "DESC" qdstring ]
   [ "OBSOLETE" whsp ]
   [ "SUP" oids ]
   [ ( "ABSTRACT" / "STRUCTURAL" / "AUXILIARY" ) whsp ]
   [ "MUST" oids ]
   [ "MAY" oids ]
   [ "X-NDS_NOT_CONTAINER" qdstrings ]
   [ "X-NDS_NONREMOVABLE" qdstrings ]
   [ "X-NDS_CONTAINMENT" qdstrings ]
   [ "X-NDS_NAMING" qdstrings ]
   [ "X-NDS_NAME" qdstrings ]
   whsp ")"
```

The following example LDIF file adds the person objectClass to the schema:

```
1 version: 1
2 dn: cn=schema
3 changetype: add
4 objectClasses: ( 2.5.6.6 NAME 'person' DESC 'Standard
5   ObjectClass' SUP ndsLoginProperties STRUCTURAL MUST
6   (cn $ sn) MAY (description $ seeAlso $ telephoneNum
7 ber $ fullName $ givenName $ initials $ uid $ userPa
8 ssword) X-NDS_NAMING ('cn' 'uid') X-NDS_CONTAINMENT
9 ('organization' 'organizationalUnit' 'domain') X-NDS
10   _NAME 'Person' X-NDS_NOT_CONTAINER '1' X-NDS_NONREMO
11   VABLE '1')
12
```

Mandatory Attributes

Mandatory attributes are listed in the MUST section of the object class description. For the person object class, the mandatory attributes are cn and sn.

Optional Attributes

Optional attributes are listed in the MAY section of the object class description. The optional attributes in the person object class are description, seeAlso, telephoneNumber, fullName, givenName, initials, uid, and userPassword.

Containment Rules

The object classes that can contain the object class being defined are given in the X-NDS_CONTAINMENT section of the object class description. The person object class can be contained by the organization, organizationalUnit, and domain object classes.

Adding a New Attribute

To add an attribute, simply add an attribute value that conforms to the specification for NDSAttributeTypeDescription to the attributes attribute of the subschemaSubentry.

```
NDSAttributeTypeDescription = "(" whsp
  numericoid whsp  ; AttributeType identifier
  [ "NAME" qdescrs ]  ; name used in AttributeType
  [ "DESC" qdstring ]  ; description
  [ "OBSOLETE" whsp ]
  [ "SUP" woid ]  ; derived from this other AttributeType
  [ "EQUALITY" woid]  ; Matching Rule name
  [ "ORDERING" woid]  ; Matching Rule name
  [ "SUBSTR" woid ]  ; Matching Rule name
  [ "SYNTAX" whsp noidlen whsp ] ;  Syntax OID
  [ "SINGLE-VALUE" whsp ]  ; default multi-valued
  [ "COLLECTIVE" whsp ]  ; default not collective
  [ "NO-USER-MODIFICATION" whsp ] ;  default user modifiable
  [ "USAGE" whsp AttributeUsage ] ;  default userApplications
  [ "X-NDS_PUBLIC_READ" qdstrings ]
                            ; default not public read ('0')
  [ "X-NDS_SERVER_READ" qdstrings ]
                            ; default not server read ('0')
  [ "X-NDS_NEVER_SYNC" qdstrings ]
                            ; default not never sync ('0')
  [ "X-NDS_NOT_SCHED_SYNC_IMMEDIATE" qdstrings ]
                        ; default sched sync immediate ('0')
```

```
[ "X-NDS_SCHED_SYNC_NEVER" qdstrings ]
                              ;  default schedule sync ('0')
[ "X-NDS_LOWER_BOUND" qdstrings ]
                              ;  default no lower bound('0')
                              ;(upper is specified in SYNTAX)
[ "X-NDS_NAME_VALUE_ACCESS" qdstrings ]
                      ; default not name value access ('0')
[ "X-NDS_NAME" qdstrings ]  ; legacy NDS name
whsp ")"
```

The following example LDIF file adds the title attribute type to the schema:

```
1 version: 1
2 dn: cn=schema
3 changetype: add
4 attributeTypes: ( 2.5.4.12 NAME 'title' DESC 'Standa
5  rd Attribute' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{
6  64} X-NDS_NAME 'Title' X-NDS_NOT_SCHED_SYNC_IMMEDIA
7  TE '1' X-NDS_LOWER_BOUND '1')
8
```

Single-Valued versus Multi-Valued

An attribute defaults to multi-valued unless it is explicitly made single-valued.
The following example LDIF file makes title single-valued by adding the
SINGLE-VALUE keyword after the SYNTAX section:

```
1 version: 1
2 dn: cn=schema
3 changetype: add
4 attributeTypes: ( 2.5.4.12 NAME 'title' DESC 'Standa
5  rd Attribute' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{
6  64} SINGLE-VALUE X-NDS_NAME 'Title' X-NDS_NOT_SCHED
7  _SYNC_IMMEDIATE '1' X-NDS_LOWER_BOUND '1')
8
```

Adding an Optional Attribute to an Existing Object Class

Although adding new schema elements is an acceptable practice, modifying
or extending existing schema elements is usually dangerous. Since every
schema element is uniquely identified by an OID, when you extend a standard
schema element, you effectively create a second definition for the element
even though it still uses the original OID. This can cause incompatibility
problems.

There are times when it is appropriate to change schema elements. For example, you might need to extend or modify new schema elements as you refine them during development. Instead of adding new attributes directly to a class, you should generally only use auxiliary classes to:

- Add new attributes to an existing object class
- Subclass an existing object class

Adding or Removing Auxiliary Classes

The following sample LDIF file creates two new attributes, creates an auxiliary class with these new attributes, and then adds an inetOrgPerson entry with the auxiliary class as an object class of the entry and with values for the auxiliary class attributes.

```
version: 1
# Add an attribute to track a bear's hair. The attribute is
# multi-valued, uses a case ignore string syntax,
# and has public read rights
# Values may include: long hair, short, curly, straight,
# none, black, and brown
# X-NDS_PUBLIC_READ '1' The 1 allows public read,
# 0 denies public read
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( 2.16.840.1.113719.1.186.4.10 NAME
'bearHair' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
X-NDS_PUBLIC_READ '1' )

# add an attribute to store a bear's picture
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( 2.16.840.1.113719.1.186.4.11 NAME
'bearPicture' SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
SINGLE-VALUE )

# create an Auxiliary class for the bearfeatures
dn: cn=schema
changetype: modify
add: objectclasses
objectclasses: (2.16.840.1.113719.1.186.6.101 NAME
'bearFeatures' MAY (bearHair $ bearPicture) AUXILIARY)

# now create a user named booboo
```

```
dn: cn=booboo,o=jellystone
changetype: add
cn: booboo
sn: bear
givenName: booboo
bearHair: Short
bearHair: Brown
bearHair: Curly
bearPicture:< file:///c:/tmp/alien.jpg
objectClass: top
objectClass: person
objectClass: inetOrgPerson
objectClass: bearFeatures

# now create a person named yogi that will later be changed
# into a bear when bearFeatures is added to its objectClass
# list
dn: cn=yogi,o=jellystone
changetype: add
cn: yogi
sn: bear
givenName: yogi
objectClass: top
objectClass: person
objectClass: inetOrgPerson

# now morph yogi into a bear by adding bearFeatures
dn: cn=yogi,o=jellystone
changetype: modify
add: objectClass
objectClass: bearFeatures
-
add: bearHair
bearHair: long
bearHair: black
#bearPicture:< file:///c:/tmp/yogi.jpg>
-

# to morph yogi back to a person, simply delete the
# objectClass bearFeatures
dn: cn=yogi,o=jellystone
changetype: modify
delete: objectClass
objectClass: bearFeatures
```

When removing auxiliary classes, you don't have to delete all of the values associated with the auxiliary class when you remove the auxiliary class from the objectClass list. eDirectory does this automatically.

If the auxiliary class had "MUST" attributes, they must all be specified in the same modify operation that adds the auxiliary class to the objectClass list, or the modification will fail.

Known Problems with XML Parsing

XML processing of any LDIF Record (LDIF format or records generated from LDAP server) will not succeed if the individual records will not satisfy all the XML rules specified in the XML file

Troubleshooting eDirectory on Linux and Solaris

This section includes information for troubleshooting eDirectory on Linux or Solaris networks.

eDirectory

Repeated eDirectory logins can use up the available memory. Disable the "Login Update" attribute using ndsimonitor to overcome this problem.

Novell Public Key Infrastructure Services

PKI Operations Not Working

If PKI operations on ConsoleOne are not working, it could be because Novell PKI Services are not running on the Linux or Solaris host. Start the PKI Services by entering `npki -l.`

If you cannot create certificates, you need to ensure that the NICI module has been properly installed. See "Initializing the NICI Module on the Server" on page 78. To verify if NICI is initialized, see "Verifying Whether NICI Is Installed and Initialized on the Server" on page 77.

LDAP Search from Netscape Address Book Fails

If you are using an export version of the Netscape browser and a KMO key size larger than 512 bits associated with the LDAP server object, the LDAP search from the Netscape Address Book might fail.

Use a domestic version of the Netscape browser in such cases.

Removing the configuration of an eDirectory server that is acting as a treekey server in a multiserver tree after having moved the existing eDirectory objects to a different server, fails with the error code for "Crucial Replica."

To complete the operation, change the Key Server DN attribute in the W0 object under Security Container > KAP to another server in the tree that has downloaded the treekey from this server.

To change the server holding the Key Server DN attribute,

1 From ConsoleOne, select the Security Object > KAP > W0.

2 Right-click W0 > select Properties > Other.

3 Select a different server for the attribute NDSPKI:SD Key Server DN.

While Uninstalling the eDirectory Server holding the CA, the KMOs created on that server will be moved to another server in the tree and become invalid

You should recreate the CA and KMOs for the tree.

We recommend that you do not uninstall the eDirectory server on which the CA for the tree has been created.

The Certificate Authority does not get created on a Solaris or Linux machine where ConsoleOne 1.3.2 is installed on the Windows ConsoleOne

To create Certificate Authority, in the dialog box to create Certificate Authority in ConsoleOne, uncheck the Allow Private Key to be Exported radio button.

NMAS on UNIX

Unable to Log in Using Any Method

After installing and configuring NMAS, restart NDS Server.

After reinstalling a method after you have uninstalled a previous instance of that method, restart NDS Server.

The User Added Using the ICE Utility is Unable to Login Using Simple Password

While adding users with simple passwords through the Novell Import Conversion Export utility, use the **-l** option.

LDAP Services

This section identifies some common problems you might experience with LDAP Services for eDirectory and how to solve them.

Ensure that the LDAP server is up before issuing a request from an LDAP client. To do so, look for the following message in the /var/nds/ndsd.log:

```
LDAP v3 for Novell eDirectory 8.6.1 started
```

For more information, see .

LDAP Clients Cannot Bind to LDAP Services for eDirectory

If an LDAP client cannot bind to LDAP Services for eDirectory, check the following:

- Is the user entering the correct username and password?
- Is the user entering an LDAP form of the name?
- Has the password expired?
- Has the server been reconfigured?

LDAP Server Isn't Using a New Configuration

Processing LDAP server configuration updates can be affected by currently bound LDAP clients.

Configuration changes are updated dynamically. The LDAP server checks for configuration changes periodically (every thirty minutes). When a change is detected, new clients cannot bind to the LDAP server during the reconfiguration process.

The LDAP server stops processing new LDAP requests for any clients currently bound and waits for any active LDAP requests to complete before updating the configuration.

LDAP operations fail when a tree is renamed using the ndsmerge utility. To work properly, the LDAP server must be refreshed or restarted after a tree is renamed.

Failure of Secure LDAP Connection

Ensure the following:

- The Certificate Authority and the Key Material object (KMO) have been created for the LDAP server.

- The KMO has been associated with the LDAP server.

- The specified CA expiration date has not elapsed. Verify whether the system date exceeds the expiration date.

- The LDAP server is listening on the secure LDAP port. The default is 636.

- SSL is enabled for LDAP server object in ConsoleOne.

For more information, see "Ensuring Secure eDirectory Operations on Linux and Solaris Systems" on page 77.

Novell Import Convert Export Utility

If an LDAP server is refreshed or unloaded, while a Novell Import Conversion Export operation is running, thee "LBURP operation is timed out" message displays on the Novell Import Conversion Export screen. The server recovers later, when the LBURP operation times out.

ndsmerge Utility

The PKI servers are not active after a merge operation. They must be restarted using the **`npki -l`** command.

Merge operations might not be successful on different versions of the product. If your server is running an older version of NDS or eDirectory, update to the latest version of eDirectory, then continue the merge operations.

The merging of two trees will not succeed if containers with similar names subordinate to a tree are present in both the source and target trees. Rename one of the containers, then continue with the merge operation.

During the graft operation, error message -611 Illegal Containment might appear. Modify the schema by running **`ndsrepair(1)`**. Run **`ndsrepair -s`**, then select Optional Schema Enhancements.

ndstrace Utility

When you turn on the **`ndstrace(1)`** screen, an error message might be displayed indicating that a primary object is invalid for the reference link. You can ignore this message if eDirectory is functioning correctly.

ndsindex Utility

While adding indexes, if you select an attribute that is not defined in the eDirectory schema, the index is displayed in the Bringing Online state forever. Remove this using the Delete option in the ndsindex utility.

ndsbackup Utility

While backing up eDirectory, "NDS Error: Connect to NDS server failed" might be displayed. This might be cause by eDirectory listening on a port other than the default port 524. Enter the port number on which eDirectory has been configured on the command line.

For example, if eDirectory is configured on port number 1524, enter the following:

```
ndsbackup sR 164.99.148.82:1524
```

Installation and Configuration

Installation Not Successful

- Check for the following error message in the /var/adm/messages directory:

  ```
  Unable to bind to SLP Multicast Address. Multicast route
      not added?
  ```

  This message is displayed if the Linux or Solaris machine is not configured for a multicast route address.

  Add the multicast route address and restart the slpuasa daemon.

- If the "-632: Error description System failure" error message appears during installation, exit from the installation process.

  Set the **n4u.base.slp.max-wait** parameter to a larger value, such as 50, in the /etc/nds.conf file, then restart the installation process.

- If you are installing eDirectory into a NetWare 5.1 tree, upgrade the eDirectory Master to NetWare 5.1 Support Pack 2a.

  For more information, see "Installing eDirectory for NetWare" on page 20.

- If you tried to upgrade an eDirectory for Solaris 2.0 installation and it was not successful, the installation might not complete the second time.

  Delete the /var/nds/.n4s_upgrade file and try the installation again.

- During installation, if the "Tree Name Not Found" error message is displayed, do the following:

  - Check whether multicast routing is enabled on the Solaris host on which you are installing the product.

  - Specify the IP address of the master server of the Tree partition.

Installation Takes a Long Time

When you are installing eDirectory into an existing tree and the installation takes a long time to complete, look at the DSTrace screen on the server. If the "-625 Transport failure" message is displayed, you need to reset the address cache.

To reset the address cache, enter the following command at the system console:

```
set dstrace = *A
```

Unable to Install into an Existing Tree over the WAN

You need a NetWare 5 or later server to install eDirectory on a Linux or Solaris system over the WAN.

Use the following procedure:

1 Enter the following command at the server console to run the Directory Agent (DA) on the NetWare server:

slpda

2 On the server containing the master replica, edit the DA_ADDR parameter in slpuasa.conf:

```
DA_ADDR =
IP_address_of_the_NetWare_server_where_the_DA_is_
running
```

3 Restart the slpuasa daemon.

4 Install eDirectory over the WAN on the Linux or Solaris system.

4a Run nds-install to add the product packages.

Do not configure the product. See "Linux and Solaris Packages for Novell eDirectory" on page 45 for more information.

4b Edit the/etc/nds.conf and add the following parameters:

```
n4u.uam.ncp-retries = 5
n4u.base.slp.max-wait = 20
```

4c Edit the /etc/slpuasa.conf to add the following parameter:

```
DA_ADDR = IP_address_of_the_NetWare_server_where_
the_DA_is_running
```

4d Run ndsconfig to configure eDirectory.

Using ndsrepair

Use the ndsrepair utility at the server console to do the following:

◆ Correct eDirectory problems such as bad records, schema mismatches, bad server addresses, and external references.

◆ Make advanced changes to the eDirectory schema.

- Perform the following operations on the eDirectory database:
  - Check the structure of the database automatically without closing the database and without database intervention.
  - Check the database index.
  - Repair the database without closing the database and locking out users.
  - Reclaim free space by discarding empty records.

Syntax

To run ndsrepair, use the following syntax:

```
ndsrepair {-U| -P| -S| -C| -E| -N| -T| -J <entry_id>} [-A
<yes/no>] [-O <yes/no>] [-F filename] [-Ad]
```

or

```
ndsrepair -R [-l <yes/no> [-u <yes/no>] [-m <yes/no>] [-i
<yes/no>] [-f <yes/no>] [-d <yes/no>] [-t <yes/no>] [-o <yes/
no>] [-r <yes/no>] [-v <yes/no>] [-c <yes/no>] [-A <yes/no>]
[-O <yes/no>] [-F filename]
```

IMPORTANT: The -Ad option should not be used without prior direction from Novell Technical Services personnel.

Table 145   ndsrepair Options

Option	Description
-U	Unattended Full Repair option. Instructs ndsrepair to run and exit without further user intervention. This is the suggested means of repair unless you are told by Novell Technical Support to perform certain operations manually. You can view the log file after the repair has completed to determine what changes ndsrepair has made.
-P	Replica and Partition Operations option. Lists the partitions that have replicas stored in the current server's eDirectory database files. The Replica options menu provides options to repair replicas, cancel a partition operation, schedule synchronization, and designate the local replica as the master replica. For more information, see "Replica and Partition Operations Option" on page 508.

Option	Description
-S	Global Schema Operations option. This option contains several schema operations that might be necessary to bring the server's schema into compliance with the master of the Tree object. However, these operations should be used only when necessary. The local and unattended repair operations already verify the schema.
-C	Check External Reference Object option. Checks each external reference object to determine if a replica containing the object can be located. If all servers that contain a replica of the partition with the object are inaccessible, the object will not be found. If the object cannot be found, a warning is posted.
-E	Report Replica Synchronization option. Reports replica synchronization status for every partition that has a replica on the current server. This operation reads the synchronization status attribute from the replica's Tree object on each server that holds replicas of the partitions. It displays the time of the last successful synchronization to all servers and any errors that have occurred since the last synchronization. A warning message is displayed if synchronization has not completed within twelve hours.
-N	Servers Known to This Database option. Lists all servers known to the local eDirectory database. If your current server contains a replica of the Tree partition, this server displays a list of all serves in the eDirectory tree. Select one server to cause the server options to be executed.
-J	Repairs a single object on the local server. You will need to provide the Entry ID (in hexadecimal format) of the object you want to repair. You can use this option instead of using the Unattended Repair (-U) option to repair one particular object that is corrupted. The Unattended Repair option can take many hours depending on the size of database. This option will help you save time.
-T	Time Synchronization option. Contacts every server known to the local eDirectory database and requests information about each server's time synchronization status. If this server contains a replica of the Tree partition, then every server in the eDirectory tree will be polled. The version of eDirectory that is running on each server is also reported.
-A	Append to the existing log file. The information is added to the existing log file. By default, this option is enabled.

Option	Description
-O	Logs the output in a file. By default, this option is enabled.
-F *filename*	Logs the output in the specified file.
-R	Repair the Local Database option. Repairs the local eDirectory database. Use the repair operation to resolve inconsistencies in the local database so that it can be opened and accessed by eDirectory. This option has suboptions that facilitate repair operations on the database. This option has function modifiers which are explained in Table 146, "Function Modifiers Used with the -R Option," on page 506.

The function modifiers used with the -R option are described below:

Table 146   **Function Modifiers Used with the -R Option**

Option	Description
-l	Locks the eDirectory database during the repair operation.
-u	Uses a temporary eDirectory database during the repair operation.
-m	Maintains the original unrepaired database.
-i	Checks the eDirectory database structure and the index.
-f	Reclaims the free space in the database.
-d	Rebuilds the entire database.
-t	Performs a tree structure check. Set it to Yes to check all the tree structure links for correct connectivity in the database. Set it to No to skip the check. The default is Yes.
-o	Rebuilds the operational schema.
-r	Repairs all the local replicas.
-v	Validates the stream files.
-c	Checks local references.

Global Schema Operations

You can use the ndsrepair -S ([-Ad] advanced switch) option to display a list showing all the schema operations that you can perform. The following table shows the available options:

Table 147   Global Schema Operation Options

Option	Description
Request Schema from Tree	Requests the master replica of the root of the tree to synchronize its schema to this server. Any changes to the schema will be propagated to this server from the master replica of the Tree object for the next 24 hours. If all servers request the schema from the master replica, network traffic can increase.
Reset Local Schema	Invokes a schema reset that clears the time stamps on the local schema and requests an inbound schema synchronization. This option is unavailable if executed from the master replica of the Tree partition. This is to ensure that all servers in the tree are not reset at the same time.
Post NetWare 5 Schema Update	Extends and modifies the schema for compatibility with Post NetWare 5 DS changes. This option requires that the server where ndsrepair is run contains a replica of the Tree partition, and that the state of the replica is On.
Optional Schema Enhancements	Extends and modifies the schema for containment and other schema enhancements. This option requires this server to contain a replica of the Tree partition, and the replica state must be On. In addition, all NetWare 4.*x* servers in the tree must have the following versions of eDirectory: ◆ NetWare 4.10 server must have NDS 5.17 or later ◆ NetWare 4.11/4.2 servers must have NDS 6.03 or later Previous versions of NDS will not be able to synchronize these changes

Option	Description
Import Remote Schema (Advanced Switch Option)	Select an eDirectory tree that contains the schema you want to add to the schema of the current tree. Once you select a tree, the server that holds the master replica of the Tree partition is contacted. The schema from that server will be used to extend the schema on the current tree.
Declare a New Epoch (Advanced Switch Option)	When you declare a new schema epoch, the master replica of the Tree partition is contacted and illegal time stamps are repaired on the schema declared on that server. All other servers will receive a new copy of the schema including the repaired time stamps. If the receiving server contains a schema that was not in the new epoch, objects and attributes that use the old schema will be changed to the Unknown object class or attribute.

Replica and Partition Operations Option

Enter the following command to display information about each replica stored on the server:

```
ndsrepair -P
```

Select the required replica. The following options are displayed:

 - Repair All Replicas

   Repairs all replicas displayed in the replica table.

 - Repair Selected Replica

   Repairs only the selected replica listed in the replica table.

   IMPORTANT: Repairing a replica consists of checking each object in the replica for consistency with the schema and data according to the syntax of the attribute. Other internal data structures associated with the replica are also checked. If you have not repaired the local eDirectory database in the last 30 minutes, you should do so before repairing any replicas.

 - Schedule Immediate Synchronization

   Schedules the immediate synchronization of all the replicas. This is useful if you are viewing the ndstrace screen and want to view eDirectory information for the synchronization process without having to wait for it to run as normally scheduled.

◆ Cancel Partition Operation

Cancels a partition operation on the selected partition. This option might be necessary if an operation appears to be incomplete or is not completing due to problems in the eDirectory tree, such as a missing server or bad communication links. Some operations might not be canceled if they have progressed too far.

◆ Designate This Server as the New Master Replica

Designates the local replica of the selected partition as the new master replica. Use this option to designate a new master replica if the original master replica is lost.

◆ Report Synchronization Status of All Servers

Reports replica synchronization status of all partitions on the current server. It displays the time of the last successful synchronization to all servers and any errors that have occurred since the last synchronization.

◆ Synchronize the Replica on All Servers

Determines the complete synchronization status on every server that has a replica of the selected partition. This helps you determine the health of a partition. If all of the servers with a replica of the partition are synchronizing properly, then the partition is considered healthy. Each server performs an immediate synchronization to every other server in the replica ring. Servers do not synchronize to themselves. Therefore, the status for the current server's own replicas is displayed as Host.

◆ Repair Ring, All Replicas

Repairs the replica ring of all the replicas displayed in the replica table.

◆ Repair Ring, Selected Replica

Repairs the replica ring of selected replica listed in the replica table.

IMPORTANT: Repairing a replica ring consists of checking the replica ring information on each server that contains a replica of a given partition and validating remote ID information. If you have not repaired the local eDirectory database in the last 30 minutes, you should do so before repairing all or selected rings. You can repair the local database using the -R option. For more information, see "-R" on page 506.

◆ View Replica Ring

Displays a list of all servers that contain a replica of the selected partition. This set of servers is called the replica ring. The replica ring list shows information about the type of replica and current status for each server in

the ring. Select a server after viewing the replica ring to view server options.

Server Options

- ◆ Report Synchronization Status on the Selected Server

  Reports replica synchronization status for a selected partition that has a replica on a selected server. This operation reads the synchronization status attribute from the replica root object on each server that holds replicas of the partitions. It displays the time of the last successful synchronization to all servers and any errors that have occurred since the last synchronization. This option displays a warning message if synchronization has not completed within twelve hours.

- ◆ Synchronize the Replica on the Selected Server

  Determines the complete synchronization status on the selected server that has a replica of the selected partition. This helps you determine the health of a partition. If the server with a replica on the partition is synchronizing properly, the partition is considered healthy. The server is immediately synchronized to every other server in the replica ring. The server does not synchronize with itself. Therefore, the status for the current server's own replica is displayed as Host.

- ◆ Send All Objects to Every Replica in the Ring

  Sends all objects from the selected server in the replica ring to all other servers that contain a replica of the partition. This operation can generate a lot of network traffic. Use this option to ensure that the selected partition's replica on the selected server in the replica ring is synchronized with all other servers in the replica ring. This operation cannot be performed on a server that contains only a subordinate reference replica of the partition.

- ◆ Receive All Objects from the Master to This Replica

  Receives all objects from the master replica to the replica on the selected servers. This operation can generate a lot of network traffic. Use this option to ensure that the selected partition's replica on the selected server in the replica ring is synchronized with the master replica. This operation cannot be performed on a server that contains only a master replica.

- ◆ View Entire Server's Name

Used to view the complete server name when the width of the server name is too long to view from within the server table.

◆ Remove This Server from Replica Ring

(Advanced switch option.) Removes a selected server from the selected replica stored on the current server. If a server appears in the replica ring but it is no longer part of the eDirectory tree or no longer contains a replica of the partition, delete the Server object using ConsoleOne. Once the Server object has been deleted, the object should eventually be excluded from the replica ring.

WARNING: Misuse of this operation can cause irrevocable damage to the eDirectory database. You should not use this option unless directed by Novell Technical Services personnel.

◆ View Entire Partition Name

Determines the complete distinguished partition name when the width of the partition is too long to view from within the replica table.

◆ Repair Time Stamps and Declare a New Epoch

(Advanced switch option.) Provides a new point of reference to the master replica so that all updates to replicas of the selected partition are current. This operation is always performed on the master replica of a partition. The master replica does not need to be in the local replica on this server. Time stamps are placed on objects when they are created or modified and must be unique. All time stamps in a master replica are examined. If any time stamps are post-dated to the current network time, they are replaced with a new time stamp.

◆ Destroy the Selected Replica on This Server

(Advanced switch option.) Removes the selected replica on this server. Using this option is not recommended. Use this option only when all other utilities are unable to delete the replica.

◆ Delete Unknown Leaf Objects

(Advanced switch option.) Deletes all objects in the local eDirectory database that have the unknown object class and maintain no subordinate objects. This option marks Unknown objects for deletion. The deletion will later be synchronized to other replicas in the eDirectory tree.

WARNING: Use this option only when the objects cannot be modified or deleted using ConsoleOne.

Options on Servers Known to This Database

The following repair options are available for servers:

* Repair All Network Address

  Checks the network address for every server in the local eDirectory database. This option searches the SLP directory agent, depending on the transport protocol available, for each server's name. Each address is then compared to the Server object's network address property and the address record of each replica property of every partition Tree object. If the addresses are different, they are updated to be the same.

* Repair Selected Server's Network Address

  Checks the network address for a specific server in the local eDirectory database files. This option searches the SLP directory agent, depending on the transport protocols currently bound for the server's name.

* View Entire Server's Name

  Displays the complete name of the server when the width of the server name is too long to view from within the server's table. This option is the same as the -P option. For more information, see "-P" on page 504.

Examples

To perform an unattended repair and log events in the /root/ndsrepair.log file, or to append events to the log file if it already exists, enter the following command:

```
ndsrepair -U  -A no -F /root/ndsrepair.log
```

To display a list of all global schema operations along with the advanced options, enter the following command:

```
ndsrepair -S -Ad
```

To repair the local database by forcing a database lock, enter the following command:

```
ndsrepair -R -l yes
```

NOTE: The input for the ndsrepair command can be redirected from an option file. The option file is a text file that can contain replica and partition operation-related options and suboptions that do not require authentication to the server. Each option or suboption is separated by a new line. Make sure that the contents of the file are in the proper sequence. If the contents are not in the proper sequence, the results will be unpredictable.

Using ndstrace

The ndstrace utility has three main parts:

- Basic functions
- Debug messages
- Background processes

Basic Functions

The basic functions of ndstrace are used to:

- View the status of the ndstrace screen in Linux or Solaris
- Initiate limited synchronization processes

To start the ndstrace screen, enter the following command at the server prompt:

`/usr/bin/ndstrace`

To initiate the basic ndstrace functions, enter commands at the server prompt using the following syntax:

`ndstrace command_option`

lists the command options that you can enter.

Table 148   **ndstace Commands**

Option	Description
ON	Starts the eDirectory trace screen with basic trace messages.
OFF	Disables the trace screen.
ALL	Starts the eDirectory trace screen and displays all the trace messages.
AGENT	Starts the eDirectory trace screen with the trace messages that are equivalent to the ON, BACKLINK, DSAGENT, JANITOR, RESNAME, and VCLIENT flags.
DEBUG	Turns on a predefined set of trace messages typically used for debugging. The flags set are ON, BACKLINK, ERRORS, EMU, FRAGGER, INIT, INSPECTOR, JANITOR, LIMBER, MISC, PART, RECMAN, REPAIR, SCHEMA, SKULKER, STREAMS, and VCLIENT.
NODEBUG	Leaves the trace screen enabled, but turns off all debugging messages previously set. This option also leaves the messages set to the ON command option.

Debugging Messages

When the ndstrace screen is enabled, the information displayed is based on a default set of filters. If you want to view more or less than the default, you can manipulate the filters using the debugging message flags. The debugging messages help you determine the status of eDirectory and verify that everything is working well.

Each eDirectory process has a set of debugging messages. To view the debugging messages on a particular process, use a plus sign (+) and the process name or option. To disable the display of a process, use a minus sign (-) and the process name or option. The following are some examples:

Table 149   **Debugging Messages**

set ndstrace = +SYNC	Enables the synchronization messages
set ndstrace = -SYNC	Disables the synchronization messages
set ndstrace = +SCHEMA	Enables the schema messages

You can also combine the debugging message flags by using the Boolean operators & (which means AND) and | (which means OR). The syntax for controlling the debugging messages at the server console is as follows:

```
set ndstrace = +trace_flag [trace_flag]
```

or

```
set ndstrace = +trace_flag> [&trace_flag]
```

Table 150 describes the trace flags for the debugging messages. You can enter abbreviations for each of the trace flags. These abbreviations or alternatives are listed within parentheses in the table.

Table 150   **Trace Flags for Debugging Messages**

Trace Flag	Description
ABUF	Messages and information related to inbound and outbound packet buffers that contain data being received in conjunction with, or in response to, an eDirectory request.
ALOC	Messages to show the details of memory allocation.
AREQ	Messages related to inbound requests from other servers or clients.
AUTH	Messages and error reports relating to authentication.
BASE	Debug error messages at the minimum debugging level.
BLNK	Backlink and inbound obituary messages and error reports.

Trace Flag	Description
CBUF	Messages related to outbound DS Client requests.
CHNG	Change cache messages.
COLL	Status and error reports concerning an object's update information when the update has been previously received.
CONN	Messages that show information about the servers your server is trying to connect to, and about errors and timeouts that might be causing your server not to connect.
DNS	Messages about the eDirectory-integrated DNS server processes.
DRLK	Distributed reference link messages.
DVRS	Messages to show DirXML™ driver-specific areas that eDirectory might be working on.
DXML	Messages to show details of DirXML events.
FRAG	Messages from the NCP fragger which breaks eDirectory messages into NCP-sized messages.
IN	Messages related to inbound requests and processes.
INIT	Messages related to the initialization of eDirectory.
INSP	Messages related to the integrity of objects in the source server's local database. Use of this flag increases the demands on the source server's disk storage system, memory, and processor. Do not leave this flag enabled unless objects are being corrupted.
JNTR	Messages related to the following background processes: janitor, replica synchronization, and flat cleaner.
LDAP	Messages related to the LDAP server.

Trace Flag	Description
LMBR	Messages related to the limber process.
LOCK	Messages related to the use and manipulation of the source server's local database locks.
LOST	Messages related to lost entries.
MISC	Messages from different sources in eDirectory.
MOVE	Messages from the move partition or move subtree operations.
NCPE	Messages to show the server receiving NCP-level requests.
NMON	Messages related to iMonitor.
OBIT	Messages from the obituary process.
PART	Messages related to partition operations from background processes and from request processing.
PURG	Messages about the purge process.
RECM	Messages related to the manipulation of the source server's database.
RSLV	Reports related to the processing of resolve name requests.
SADV	Messages related to the registration of tree names and partitions with Service Location Protocol (SLP).
SCMA	Messages related to the schema synchronization process.
SCMD	Messages showing the details of Schema-related operations. It gives details of both inbound and outbound synchronization.
SKLK	Messages related to the replica synchronization process.

Trace Flag	Description
SPKT	Messages related to eDirectory NCP server-level information.
STRM	Messages related to the processing of attributes with a Stream syntax.
SYDL	Messages showing more details during the replication process.
SYNC	Messages about inbound synchronization traffic (what is being received by the server).
TAGS	Displays the tag string that identifies the trace option that generated the event on each line displayed by the trace process.
THRD	Messages to show when any background processes (threads) begin and end.
TIME	Messages about the transitive vectors that are used during the synchronization process.
TVEC	Messages related to the following attributes: Synchronize Up To, Replica Up To, and Transitive Vector.
VCLN	Messages related to the establishment or deletion of connections with other servers.

As you use the debugging messages in ndstrace, you will find that some of the trace flags are more useful than others. One of the favorite ndstrace settings of Novell Technical Services™ is actually a shortcut:

```
set ndstrace = A81164B91
```

This setting enables a group of debugging messages.

Background Processes

In addition to the debugging messages, which help you check the status of eDirectory, there is a set of commands that forces the eDirectory background processes to run. To force the background process to run, place an asterisk (*) before the command, for example:

```
set ndstrace = *H
```

You can also change the status, timing, and control for a few of the background processes. To change these values, place an exclamation point (!) before the command and enter a new parameter or value, for example:

```
set ndstrace = !H 15 (parameter_value_in_minutes)
```

The following is the syntax for each statement controlling the background processes of eDirectory:

```
set ndstrace = *trace_flag [parameter]
```

or

```
set ndstrace = !trace_flag [parameter]
```

Table 151 lists the trace flags for the background processes, any required parameters, and the process the trace flags will display.

Table 151   **Trace Flags for Background Processes**

Trace Flag	Parameters	Description
*A	None	Resets the address cache on the source server.
*AD	None	Disables the address cache on the source server.
*AE	None	Enables the address cache on the source server.
*B	None	Schedules the back link process to begin execution on the source server in one second.
!B	Time	Sets the interval, in minutes, for the back link process. The default interval is 1500 minutes (25 hours). The range is 2 to 10080 minutes (168 hours).
*CT	None	Displays the source server's outbound connection table and the current statistical information for the table. These statistics do not give any information about the inbound connections from other servers or clients to the source server.
*CTD	None	Displays, in comma-delimited format, the source server's outbound connection table and the current statistical information for the table. These statistics do not give any information about the inbound connections from other servers or clients to the source server.

Trace Flag	Parameters	Description
*D	Replica rootEntry ID	Removes the specified local entry ID from the source server's Send All Object list. The entry ID must specify a partition root object that is specific to the server's local database. This command is usually used only when a Send All Updates process is endlessly trying to show updates and failing because a server is inaccessible.
!D	Time	Sets the inbound and outbound synchronization interval to the specified number of minutes. The default interval is 24 minutes. The range is 2 to 10080 minutes (168 hours).
!DI	Time	Sets the inbound synchronization interval to the specified number of minutes. The default interval is 24 minutes. The range is 2 to 10080 minutes (168 hours).
!DO	Time	Sets the outbound synchronization interval to the specified number of minutes. The default interval is 24 minutes. The range is 2 to 10080 minutes (168 hours).
*E	None	Re-initializes the source server's entry cache.
!E	None	Schedules the inbound and outbound synchronization processes to begin execution.
!EI	None	Schedules the inbound synchronization process to begin execution.
!EO	None	Schedules the outbound synchronization process to begin execution.
*F	None	Schedules flat cleaner process, which is part of the janitor process, to begin execution on the source server in five seconds.
!F	Time	Sets the interval, in minutes, for the flat cleaner process. The default interval is 240 minutes (4 hours). The range is 2 to 10080 minutes (168 hours).
*G	Replica rootEntry ID	Rebuilds the change cache of the specified root partition ID.

Trace Flag	Parameters	Description
*H	None	Schedules the replica synchronization process to begin execution immediately on the source server.
!H	Time	Sets the interval, in minutes, for the heartbeat synchronization process. The default interval is 30 minutes. The range is 2 to 1440 minutes (24 hours).
*HR	None	Clears the in-memory last sent vector.
*I	Replica rootEntry ID	Adds the specified local entry ID to the source server's Send All Object list. The entry ID must specify a partition root object that is specific to the server's local database. The replica synchronization process checks the Send All Object list. If the entry ID of a partition's root object is in the list, eDirectory synchronizes all objects and attributes in the partition, regardless of the value of the Synchronized Up To attribute.
!I	Time	Sets the interval, in minutes, for the heartbeat synchronization process. The default interval is 30 minutes. The range is 2 to 1440 minutes (24 hours).
*J	None	Schedules the purge process, which is part of the replica synchronization process, to begin running on the source server.
!J	Time	Sets the interval, in minutes, for the janitor process. The default interval is 2 minutes. The range is 1 to 10080 minutes (168 hours).
*L	None	Schedules the limber process to begin running on the source server in five seconds.
*M	Bytes	Changes the maximum file size used by the source server's NDSTRACE.LOG file. The command can be used regardless of the state of the debug file. The <bytes> specified must be a hexadecimal value between 10000 bytes and 100MB. If the value specified is higher or lower than the specified range, no change occurs.
!M	None	Reports the maximum memory used by eDirectory.

Trace Flag	Parameters	Description
!N	0|1	Sets the name form: zero (0) specifies hex only, and one (1) specifies full dot form.
*P	None	Displays the tunable parameters and their default settings.
*R	None	Resets the TTF file, which is the SYS:SYSTEM\NDSTRACE. DBG file by default. This command is the same as the SET parameter NDS Trace File Length Set to Zero.
*S	None	Schedules the Skulker process, which checks whether any of the replicas on the server need to be synchronized.
!SI	Time	Sets the interval, in minutes, for the inbound schema synchronization process. The default interval is 24 minutes. The range is 2 to 10080 minutes (168 hours).
!SO	Time	Sets the interval, in minutes, for the outbound schema synchronization process. The default interval is 24 minutes. The range is 2 to 10080 minutes (168 hours).
!SIO	Time	Disables the inbound schema synchronization process for the specified number of minutes. The default interval is 24 minutes. The range is 2 to 10080 minutes (168 hours).
!SO0	Time	Disables the inbound schema synchronization process for the specified number of minutes. The default interval is 24 minutes. The range is 2 to 10080 minutes (168 hours).
*SS	None	Forces immediate schema synchronization.
*SSA	None	Schedules the schema synchronization process to begin immediately and forces schema synchronization with all target servers, even if they have been synchronized in the last 24 hours.

Trace Flag	Parameters	Description
*SSD	None	Resets the source server's Target Schema Sync list. This list identifies which servers the source server should synchronize with during the schema synchronization process. A server that does not hold any replicas sends a request to be included in the target list of a server that contains a replica with its server object.
*SSL	None	Prints the schema synchronization list of target servers.
*ST	None	Displays the status information for the background processes on the source server.
*STX	None	Displays the status information for the backlink process (external references) on the source server.
*STS	None	Displays the status information for the schema synchronization process on the source server.
*STO	None	Displays the status information for the backlink process (obituaries) on the source server.
*STL	None	Displays the status information for the limber process on the source server.
!T	Time	Sets the interval, in minutes, for checking the server's UP state. The default interval is 30 minutes. The range is 1 to 720 minutes (12 hours).
*U	Optional ID of server	If the command does not include an entry ID, changes the status of any server that has been previously labeled "down" to "up." If the command includes a local entry ID, changes the status of the specified server from "down" to "up." Entry IDs are specific to the source server's database and must refer to an object that represents a server.
!V	A list	Lists the restricted eDirectory versions. If no versions are listed, there are no restrictions. Each version is separated by a comma.
*Z	None	Displays the currently scheduled tasks.

Troubleshooting Dynamic Groups

In this release of eDirectory, you should use LDAP to manage Dynamic Group objects. There are no ConsoleOne snap-ins available with eDirectory 8.6.1 to manage dynamic groups. Also, third-party products such as Netscape's admin console cannot be used to manage Novell eDirectory dynamic groups because the schema and the functionality provided differs from Netscape's implementation.

Configuring the memberQueryURL Attribute

The memberQueryURL attribute is defined as a multi-valued attribute to enable future enhancements, but in eDirectory 8.6.1, only the first value of the multi-valued attribute is used for the dynamic member expansion. To avoid confusion, set only one value for the memberQueryURL attribute, and use the "replace:" option instead of "add:" to specify a query URL value for the memberQueryURL attribute.

Changing treeName or Search baseDN

Referential integrity is not supported for distinguished names embedded in the memberQueryURL attribute. If a dynamic group object is created in a tree and the treeName or the search baseDN changes subsequently, the dynamic members will not be visible. You can correct this by reading the memberQueryURL attribute and correcting the treeName or the baseDN. The same thing applies for attribute names or class names that are in the search filter and subsequently removed.

Configuring the dgIdentity Attribute

A dgIdentity attribute on the Dynamic Group object can be set to the distinguished name of an entry whose credentials and rights should be used to expand the dynamic members of the group. The dgIdentity entry should always be chosen so that it is on the same partition as the Dynamic Group object. If it is not on the same partition, the dynamic members will not be visible. If no dgIdentity attribute is specified, the expansion of dynamic members will bind as public (anonymous). In that case, [Public] should at least have Read/Compare rights on all the attribute that occur in the search filter in the memberQueryURL, and also have Browse rights under the baseDN specified in the memberQueryURL.

Creating Dynamic Group Objects on a pre-eDirectory 8.6.1 server

If Dynamic Group objects are created on a pre-eDirectory 8.6.1 server, either by extending the schema to the eDirectory 8.6.1 schema or by adding the server to a tree that contains eDirectory 8.6.1 servers, the Dynamic Group object will be created but its dynamic members will not be visible on the pre-eDirectory 8.6.1 server. If the server is subsequently upgraded to eDirectory 8.6.1, the dynamic members will still not be visible until the Dynamic Group objects are upgraded.

Troubleshooting ConsoleOne

For more information on troubleshooting ConsoleOne, see "Troubleshooting" in *ConsoleOne User Guide*.

Unable to Browse the eDirectory Tree

You cannot browse the eDirectory tree if:

- The Tree object in the eDirectory tree has been deleted, renamed, or moved. Logout from the network and log in to the tree again to view the changes.

- The ndsd daemon is down. Restart the daemon by entering one of the following commands:

    - On Linux systems, type: **/etc/rc.d/init.d/ndsd start**

    - On Solaris systems, type: **/etc/init.d/ndsd start**

- The following error message appears:

```
An attempt to resolve SVC (switched virtual circuit)
  failed.
```

This is because the tree is the primary tree but the server is not the primary server. Set the server as the primary server.

If you perform any one of the actions listed above, open a new ConsoleOne window to browse the tree.

Running Pure IP Environments on NetWare Servers

On NetWare servers, in a pure IP environment, ConsoleOne does not see the eDirectory tree of the server that ConsoleOne is running on.

To fix this, make the following changes in the AUTOEXEC.NCF file:

- ◆ Add the statement **LOAD SCMD.NLM** after the LOAD and BIND statements for TCPIP.

- ◆ Add IPX to the serverID.

- ◆ Then restart the server.

For more information on this issue, refer to Solution #2943528 (http://support.novell.com) in the Novell Knowledgebase.

A NMAS Considerations

This appendix discusses the implications of merging two trees that have Security containers installed in one or both of the trees.

Setting Up a Security Container as a Separate Partition

Novell® Modular Authentication Services™ (NMAS™) relies on the storage of policies that are global to the Novell eDirectory™ tree. The eDirectory tree is effectively the security domain. The security policies must be available to all servers in the tree.

NMAS places the authentication policies and login method configuration data in the Security container that is created in the Tree object in NetWare® 5.*x* or later eDirectory trees. This information must be readily accessible to all servers that are enabled for NMAS. The purpose of the Security container is to hold global policies that relate to security properties such as login, authentication, and key management.

With NMAS, we recommend that you create the Security container as a separate partition, and that the container be widely replicated. This partition should only be replicated as a read/write partition on those servers in your tree that are highly trusted.

IMPORTANT: Because the Security container will contain global policies, be careful where writable replicas are placed, since these servers can modify the overall security policies specified in the eDirectory tree. In order for users to log in with NMAS, replicas of the User objects must be on the NMAS server.

Merging Trees with Multiple Security Containers

Special considerations are needed when merging eDirectory trees where a Security container has been installed in one or both of the trees. Make sure that this is something you really want to do. This procedure can be a time-consuming and laborious task.

IMPORTANT: These instructions are complete for trees with Novell Certificate Server™ 2.02 and earlier, Novell Single Sign-on 1.*x*, and NMAS 1.*x*.

To merge trees with multiple security containers:

1 In ConsoleOne™, identify the trees that will be merged.

2 Identify which tree will be the source tree and which tree will be the target tree.

Keep in mind these security considerations when choosing which tree will be the source and which tree will be the target:

- ◆ Any certificates signed by the source tree's Organizational CA must be deleted.

- ◆ The source tree's Organizational CA must be deleted.

- ◆ All user secrets stored in Secret Store on the source tree must be deleted.

- ◆ All NMAS login methods in the source tree must be deleted and reinstalled in the target tree.

- ◆ All NMAS users that were in the source tree must be reenrolled when the trees are merged.

- ◆ All users and servers that were in the source tree must have new certificates created for them when the trees are merged.

- ◆ All users that were in the source tree must have their secrets reinstalled into their Secret Store.

If neither the source tree nor the target tree has a container named Security under the root of the tree, or if only one of the trees has the Security container, no further action is required. Otherwise, continue with the remaining procedures in this section.

Product-Specific Operations to Perform Prior to a Tree Merge

Novell Certificate Server

If Novell Certificate Server has been installed on any server in the source tree, you should complete the following steps.

Depending on how the product was used, the objects and items referred to might not be present. If the objects and items referred to in a given step are not present in the source tree, skip the step.

NOTE: Previous versions of Novell Certificate Server were called Public Key Infrastructure Services (PKIS).

1 Any Trusted Root certificates in the source tree should be installed in the target tree.

   Trusted Root certificates are stored in Trusted Root objects, which are contained by Trusted Root containers. Trusted Root containers can be created anywhere within the tree; however, only the Trusted Root certificates that are in the Trusted Root containers within the Security container must be moved manually from the source tree to the target tree.

2 Install the Trusted Root certificates in the target tree.

   2a Pick a Trusted Root container in the Security container in the source tree.

   2b Create a Trusted Root container in the Security container of the target tree with the exact name used in the source tree (Step 2a).

   2c In the source tree, open a Trusted Root object in the selected Trusted Root container and export the certificate.

   IMPORTANT: Remember the location and filename you choose; you will use them in the next step.

   2d In the target tree, create a Trusted Root object in the container that you created in Step 2b. Specify the same name as the source tree and, when prompted for the certificate, specify the file that you created in Step 2c.

   2e Delete the Trusted Root object in the source tree.

   2f Repeat Step 2c through Step 2e until all Trusted Root objects in the selected Trust Root container have been installed into the target tree.

   2g Delete the Trusted Root container in the source tree.

   2h Repeat Step 2a through Step 2g until all Trusted Root containers have been deleted in the source tree.

3 Delete the Organizational CA in the source tree. The Organizational CA object is in the Security container.

Any certificates signed by the Organizational CA of the source tree will be unusable following Step 3. This includes server certificates and user certificates that have been signed by the Organizational CA of the source tree.

4 Delete every Key Material object in the source tree that has a certificate signed by the Organizational CA of the source tree.

Key Material objects in the source tree with certificates signed by other CAs will continue to be valid and do not need to be deleted.

If you are uncertain about the identity of the signing CA for any Key Material object, reference the Trusted Root Certificate section of the Certificates tab in the Key Material object properties page.

5 Delete all user certificates in the source tree that have been signed by the Organizational CA of the source tree.

If users in the source tree have already exported their certificates and private keys, those exported certificates and keys will continue to be usable. Private keys and certificates that are still in eDirectory will no longer be usable after you perform Step 3 on page 530.

For each user with certificates, open the Properties of the User object. Under the Certificates section of the Security tab, a table will list all the certificates for the user. All of those certificates with the Organizational CA as the issuer must be deleted.

User certificates will only be present in the source tree if Novell Certificate Server 2.0 or later has been installed on the server that hosts the Organizational CA in the source tree.

Novell Single Sign-On

If Novell Single Sign-on has been installed on any server in the source tree, you should complete the following step.

Depending on how the product was used, the objects and items referred to might not be present. If the objects and items referred to are not present in the source tree, skip the step.

1 Delete all Novell Single Sign-on secrets for users in the source tree.

For every user using Novell Single Sign-on in the source tree, open the Properties of the User object. All of the user's secrets will be listed under the Secret Store section of the of the Security tab. Delete all listed secrets.

NMAS

If NMAS has been installed on any server in the source tree, you should complete the following steps.

Depending on how the product was used, the objects and items referred to might not be present. If the objects and items referred to are not present in the source tree, skip the step.

1 In the target tree, install any NMAS login methods that were in the source tree but not in the target tree.

To ensure that all of the necessary client and server login components are properly installed in the target tree, we recommend that you install new login methods using original Novell or vendor-supplied sources.

Although methods can be reinstalled from existing server files, establishing a clean installation from Novell or vendor-supplied packages is usually simpler and more reliable.

2 To ensure that the previously established login sequences in the source tree are available in the target tree, migrate the desired login sequences.

2a In ConsoleOne, select the Security container in the source tree.

2b Right-click the Login Policy object and select Properties.

2c For each login sequence listed in the Defined Login Sequences drop-down menu, notate the Login Methods used (listed in the right window).

2d Select the Security container in the target tree and replicate the login sequences using the same login methods notated in Step 2c.

2e Click OK when you are finished.

3 Delete NMAS login security attributes in the source tree.

3a In the Security container of the source tree, delete the Login Policy object.

3b In the Authorized Login Methods container of the source tree, delete all login methods.

3c Delete the Authorized Login Methods container in the source tree.

Novell Security Domain Infrastructure

If Novell Certificate Server, Novell Single Sign-on, or NMAS has been installed on any server in the source tree, the Novell Security Domain Infrastructure (SDI) will be installed. If SDI has been installed, complete the following steps.

Depending on how the product was used, the objects and items referred to might not be present. If the objects and items referred to are not present in the source tree, skip the step.

1 Delete the W0 object and then the KAP container in the source tree.

The KAP container is in the Security container. The W0 object is in the KAP container.

2 On all servers in the source tree, delete the Security Domain Infrastructure (SDI) keys by deleting the SYS:\SYSTEM\NICI\NICISDI.KEY file.

IMPORTANT: Make sure that you delete this file on *all* servers in the source tree.

Other Security-Specific Operations

If a Security container exists in the source tree, delete the Security container before you merge the trees.

Performing the Tree Merge

eDirectory trees are merged using the ndsmerge utility. For more information, refer to the DSMERGE documentation (http://www.novell.com/documentation/lg/nds73/docui/index.html#../maintenu/data/hpqtpzmg.html).

Product-Specific Operations to Perform after the Tree Merge

Novell Security Domain Infrastructure

If the W0 object existed in the target tree before the merge, the Security Domain Infrastructure (SDI) keys used by the servers that formerly resided in the target tree must be installed in the servers that formerly resided in the source tree.

The easiest way to accomplish this is to install Novell Certificate Server 2.0 or later on all servers formerly in the source tree that held SDI keys (the SYS:\SYSTEM\NICI\NICISDI.KEY file). This should be done even if the Novell Certificate Server has already been installed on the server.

If the W0 object did not exist in the target tree before the merge but did exist in the source tree, the SDI must be reinstalled in the resulting tree.

The easiest way to accomplish this is to install Novell Certificate Server v2.0 or later on the servers in the resulting tree. Novell Certificate Server must be installed on the servers formerly in the source tree that held SDI keys (the SYS:\SYSTEM\NICI\NICISDI.KEY file). It can also be installed on other servers in the resulting tree.

Novell Certificate Server

If you are using Novell Certificate Server, after the tree merge:

1 Reissue certificates for servers and users that were formerly in the source tree, as necessary.

We recommend that you install Novell Certificate Server 2.0 or later on all servers that hold a replica of the partition containing a User object.

In order to issue a certificate for a server, Novell Certificate Server 2.0 or later must be installed.

Novell Certificate Server 2.0 or later must be installed on the server that hosts the Organizational CA.

Novell Single Sign-On

If you are using Novell Single Sign-on, after the tree merge:

1 Re-create Secret Store secrets for users that were formerly in the source tree, as necessary.

NMAS

If you are using NMAS, after the tree merge:

1 Reenroll NMAS users that were formerly in the source tree, as necessary.