

NIS Extension Guide

Netscape Directory Server

Version 4.11

Copyright © 1999 Sun Microsystems, Inc. Some preexisting portions Copyright © 1999 Netscape Communications Corp. All rights reserved.

Sun, Sun Microsystems, the Sun Logo, Java, HotJava and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. Netscape and the Netscape N logo are registered trademarks of Netscape Communications Corporation in the U.S. and other countries. Other Netscape logos, product names, and service names are also trademarks of Netscape Communications Corporation, which may be registered in other countries.

This product was derived in part from the Lightweight Directory Access Protocol (LDAP) software that was developed at the University of Michigan and is copyright (c) 1992-1996 Regents of the University of Michigan. All rights reserved.

Copyright (C) 1992-1999 Lucent Technologies Inc. All rights reserved.

Federal Acquisitions: Commercial Software -- Government

Users Subject to Standard License Terms and Conditions

The product described in this document is distributed under licenses restricting its use, copying, distribution, and decompilation. No part of the product or this document may be reproduced in any form by any means

without prior written authorization of the Sun-Netscape Alliance and its licensors, if any.

THIS DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Part Number: 806-4251-01



Recycled and Recyclable Paper

Contents

Netscape Directory Server 4.11 Overview	13
Prerequisite Reading	13
What Is in This Book?	13
Conventions Used in This Book	14
Chapter 1 Introduction to Synchronized NIS/LDAP Service	15
Benefits of Netscape Directory Server	16
NIS/LDAP Compared Terminology	16
Transitioning to Synchronized Operation.....	17
Daemons Providing Synchronized Service.....	20
dsypserv	21
dsyppasswdd	22
dsypxfr	22
dsmakedbm	22
dsypinit.....	22
dsypsync.....	23
dsyp	23
dsypaddmap	23
dsypdelmap	23
Chapter 2 Initializing and Operating the NIS Service	25
Initializing Synchronized Operation	25
Setting Up an NIS Server.....	26
Setting up Synchronized Service	26
Running the dsypinstall Script	27
Verifying your NIS Installation	28
Checking NIS Information	28
Checking Server Status	29
Getting the List of Supported Maps	29

Access Control on NIS Information	30
Configuring the NIS Service.....	31
Setting NIS Configuration Variables	31
Configuring NIS Subtrees	32
Updating NIS Maps	33
Propagating NIS Maps.....	34
Standard NIS Replication.....	34
LDAP Replication	34
Adding and Deleting NIS Maps.....	35
Chapter 3 Using Deja to Update NIS Information.....	37
Introduction to Deja	38
Starting Deja	40
Logging In	41
General Operations	42
Setting the Display Options	43
Setting Deja Properties.....	43
User Properties	43
Connection Properties	44
Opening a New Deja Window	44
Closing a Deja Window	44
Reconnecting Deja to the Directory Server	45
Connecting to Another Directory Server	45
Refreshing the Browser Window	45
Operations on NIS Entries	46
Viewing an Entry	46
The View Window	47
Closing a View Window	47
Copying an Entry From a View Window.....	47
Highlighting an Entry From a View Window	47
Creating a New Entry.....	47
Naming an Entry	49
Selecting Object Classes	50
Selecting Attributes	50

Cancel	53
Deleting an Entry	53
Cut, Copy and Paste.....	54
Cutting an Entry	54
Copying an Entry	54
Pasting an Entry	55
Restoring an Entry	56
Modifying an Entry.....	56
Renaming an Entry	56
Searching for an Entry	58
Search Results List.....	59
Setting Deja Properties	59
File Structure	59
File Syntax	60
Labels	60
General Properties	61
Standard LDAP Properties	62
Hiding Attributes	62
Login Parameters	63
NIS Properties.....	64
NIS_MAPS	64
NIS_FILTER. <i>map.name</i>	64
NIS_DOMAIN. <i>map.name</i>	64
NIS_NAMINGATTR. <i>map.name</i>	64
NIS_ROOT. <i>map.name</i>	65
NIS_OCLASS. <i>map.name</i>	65
NIS_LIST. <i>map.name</i>	65
NIS_ADD. <i>map.name</i>	65
NIS_LIST.default	65
Adding a NIS Map to Deja using <i>dejasync</i>	66
Default NIS Map Definitions.....	67

Chapter 4 NIS Information in the LDAP Directory	71
NIS Files/LDAP Subtrees	71
NIS File Entries/LDAP Entries.....	72
Generic Mapping.....	74
aliases Mapping	74
bootparams Mapping	76
ethers Mapping	77
group Mapping.....	78
hosts Mapping.....	78
netgroup Mapping	79
networks Mapping	80
passwd Mapping	80
protocols Mapping	81
rpc Mapping.....	81
ypservers Mapping	82
NIS Schema	82
NIS Object Classes.....	83
automount.....	83
bootableDevice (auxiliary object class)	83
ieee802Device (auxiliary object class).....	84
ipHost (auxiliary object class).....	84
ipNetwork.....	84
ipProtocol	84
nisMailAlias	85
nisMap.....	85
nisNetGroup	85
nisNetId	85
nisObject.....	86
nisSunObject	86
oncRpc.....	86
posixAccount (auxiliary object class)	86
posixGroup	87
shadowAccount (auxiliary object class).....	87

sunNisMap	87
sunNisServer	88
NIS Attributes	88
automountInformation	88
bootFile	89
bootParameter	89
commonName	89
description	89
gecos	89
gidNumber	89
homeDirectory	90
ipHostNumber	90
ipNetmaskNumber	90
ipNetworkNumber	90
ipProtocolNumber	90
localityName	90
loginShell	91
macAddress	91
manager	91
memberNisNetgroup	91
nisMapEntry	91
nisMapName	91
nisNetgroupTriple	92
nisNetIdGroup	92
nisNetIdHost	92
nisNetIdUser	92
oncRpcNumber	92
rfc822MailMember	92
seeAlso	93
serialNumber	93
shadowLastChange	93
shadowExpire	93
shadowFlag	93

shadowInactive	93
shadowMax	94
shadowMin	94
shadowWarning	94
sunNisDbmCache	94
sunNisDnsForwarding	94
sunNisDomain	95
sunNisInputFile	95
sunNisKey	95
sunNisLoadMap	95
sunNisMapFullName	95
sunNisMapState	96
sunNisMaster	96
sunNisOutputName	96
sunNisSecurityMode	96
uidNumber	96
userid	97
userPassword	97
NIS/LDAP Mapping Summary	97
Chapter 5 NIS Command & File Reference	99
Deja.properties	99
Synopsis	99
Description	99
File Structure	100
File Syntax	100
Labels	100
User Input	101
General Properties	101
Standard LDAP Properties	102
NIS Properties	104
RADIUS Properties	106
See Also	108

dejasync	108
Synopsis	108
Description.....	109
nis.mapping File	109
radius.mapping File	109
Options.....	110
See Also	110
dsexport	110
Synopsis	110
Description.....	111
Options.....	111
See Also	113
dsimport	114
Synopsis	114
Description.....	114
Options.....	114
See Also	117
dsyp	118
Synopsis	118
Description.....	118
Options.....	118
See Also	118
dsypaddmap	119
Synopsis	119
Description.....	119
Options.....	119
See Also	120
dsypdelmap	120
Synopsis	120
Description.....	121
Options.....	121
See Also	121

dsypinit.....	122
Synopsis	122
Description	122
Options	122
Client Option	123
Master Server Options.....	123
Slave Server Options	124
Disable Options	124
See Also	125
dsypinstall	125
Synopsis	125
Description	125
Options	126
See Also	126
dsypasswdd	126
Synopsis	126
Description	127
Options	127
Notes	128
See Also	128
dsypsync.....	128
Synopsis	128
Description	129
Options	129
See Also	129
dsypxfr	129
Synopsis	129
Description	130
Options	131
See Also	131
nis.at.conf.....	132
Synopsis	132
Description	132

See Also	132
nis.conf	133
Synopsis	133
Description.....	133
See Also	133
nis.mapping	134
Synopsis	134
Description.....	134
Configuration Parameters	134
Mapping Information	135
See Also	136
nis.oc.conf.....	136
Synopsis	136
Description.....	136
See Also	137
Appendix A Mapping Syntax and Semantics	139
File Structure	139
Mapping Semantics	140
Common Section	142
BASE_DN	142
MAP_NAME	143
PRIVATE_OBJECTCLASSES	143
Dynamic Section.....	143
LINE	143
MATCH_FILTER	144
ALL_FILTER	144
DC_NAMING	144
Export Section	144
Import Section	145
Mapping Syntax.....	145
Common	146
Dynamic.....	146
Extract.....	146

Condense	147
split Function	148
string2instances Function	148
instances2string Function	149
trim Function	150
getrdn Function	150
exclude Function	150
Build.....	151
Index.....	153

Introduction

The NIS Extension Guide explains how to use the Netscape Directory Server as an NIS server. It explains how to initialize and configure the NIS service to use the LDAP directory to store information. It also describes how NIS information is stored in the LDAP directory.

Netscape Directory Server 4.11 Overview

The NIS Extension provided with Solaris™ Extensions for Netscape Directory Server 4.11 offers all the tools required to operate an NIS service that is synchronized with the directory server.

Prerequisite Reading

For information on how to configure and manage the directory server and the directory contents, refer to the *Netscape Directory Server Administrator's Guide*. For basic directory and architectural concepts, refer to the *Netscape Directory Server Deployment Manual*.

Instructions for installing the Netscape Directory Server components are contained in the *Netscape Directory Server Installation Guide*.

For information on NIS administration, refer to the documentation for your Solaris™ operating environment.

What Is in This Book?

This book explains how to set up and manage the synchronization of an NIS naming service with the Netscape Directory Server. It is intended for NIS administrators who want to move to an LDAP-based NIS service. It is assumed that you are familiar with NIS administration.

Conventions Used in This Book

This section explains the conventions used in this book.

Monospaced font—This typeface is used for any text that appears on the computer screen or text that you should type. It is also used for filenames, functions, and examples.

Note Notes and Warnings mark important information. Make sure you read the information before continuing with a task.

|—The vertical bar is used as a separator for user interface elements. For example, Configuration|Logs means you should go to the Configuration tab on the Directory Server Console and then select the Logs icon.

Throughout this book you will see path references of the form

`<NSHOME>/slapd-<serverID>/...`

In these situations, `<NSHOME>` represents the directory where you installed the server, and `<serverID>` represents the server identifier you gave the server when you installed it. For example, if you installed your server in `/export/ns-home` and gave the server an identifier of `phonebook`, then the actual path would be

`/export/ns-home/slapd-phonebook/...`

Introduction to Synchronized NIS/LDAP Service

The Network Information Service (NIS) provides a robust and reliable naming service that holds information on network users, groups, host machines, servers, and generally all information necessary to operate a network. However, with the growing acceptance of LDAP directories, companies find that they also want to hold the same type of information in their LDAP directory.

This chapter provides an introduction to using the NIS name service in conjunction with the Netscape Directory Server. All NIS information is stored in the LDAP directory to provide a single repository for network information. All of the features provided by the standard NIS environment are preserved.

This chapter explains the stages in transitioning from a standard NIS service to a naming service synchronized with the Netscape Directory Server.

This chapter contains the following sections:

- “Benefits of Netscape Directory Server” on page 16
- “NIS/LDAP Compared Terminology” on page 16
- “Transitioning to Synchronized Operation” on page 17
- “Daemons Providing Synchronized Service” on page 20

Benefits of Netscape Directory Server

Using the Netscape Directory Server to store NIS information provides several benefits:

- In a standard NIS environment, information is stored in a set of flat files, referred to as the NIS source files, or the *etc* files, because their default location is under the `/etc` directory. When LDAP synchronization is enabled, this information is stored in the LDAP directory which serves as a common repository for information.
- In a standard NIS environment, when information is propagated from master servers to slave servers, entire tables (or maps) are propagated, not just the updates. With LDAP replication, it is possible to replicate just the changed information to increase efficiency.
- The information stored in the Netscape Directory Server is accessible through the LDAP protocol and can potentially be shared with all applications and users in the organization, although you can restrict access if necessary.

NIS/LDAP Compared Terminology

In an NIS environment, information is stored in *tables*, which are also often referred to as *maps*. These tables are stored on disk, and are constructed from the NIS source files. The source file and corresponding table are identified using the same name which identifies the type of information they hold. For example, the `/etc/hosts` file contains a list of host names and their corresponding IP addresses which are used to build the hosts table.

In LDAP, information is stored in *entries*. The type of entry, or *object class*, and the mandatory and optional attributes associated with that object class, identify the type of information held in the entry. For example, the object class `nisObject` is used to create an LDAP entry that represents a line in an NIS file. A mandatory attribute of `nisObject` is `nisMapName`.

Detailed examples of how the contents of NIS files are mapped onto LDAP entries are provided in Chapter 4, “NIS Information in the LDAP Directory.”

In an NIS environment where multiple servers hold copies of the NIS tables, the server that holds the reference copy is called the *master* server, and all servers that hold copies are called *slave* servers. The operation of copying information between the master and the slave servers is called *propagation*.

In an LDAP environment, you might see the terms *master* and *replica* servers. However, in the Netscape Directory Server documentation, a master server is referred to as a *supplier*, and a slave server as a *consumer*. The operation of copying information between a supplier and the consumers is called *replication*.

Transitioning to Synchronized Operation

If you have a well established NIS environment, the best way to manage the transition without disrupting your naming service is outlined in Figure 1.1, Figure 1.2, and Figure 1.3. The procedure for initializing synchronized NIS/LDAP service is described in Chapter 2, “Initializing and Operating the NIS Service.”

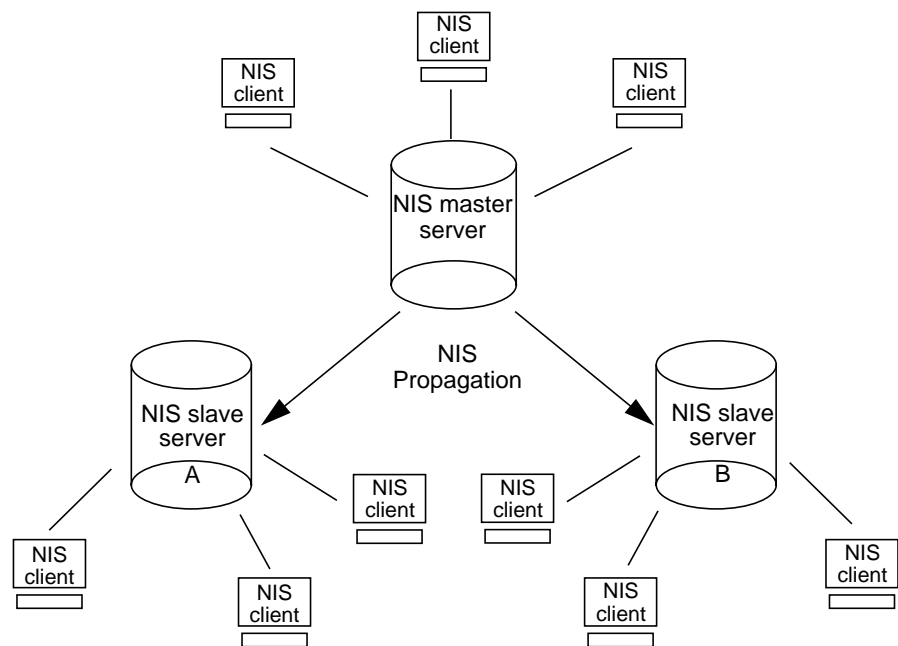


Figure 1.1 Pure NIS environment

Figure 1.1 shows a pure NIS environment with NIS requests from clients handled by the closest NIS server on the network. The synchronization of information held on master and slave servers is handled through NIS propagation of tables, using the standard `yppush` process.

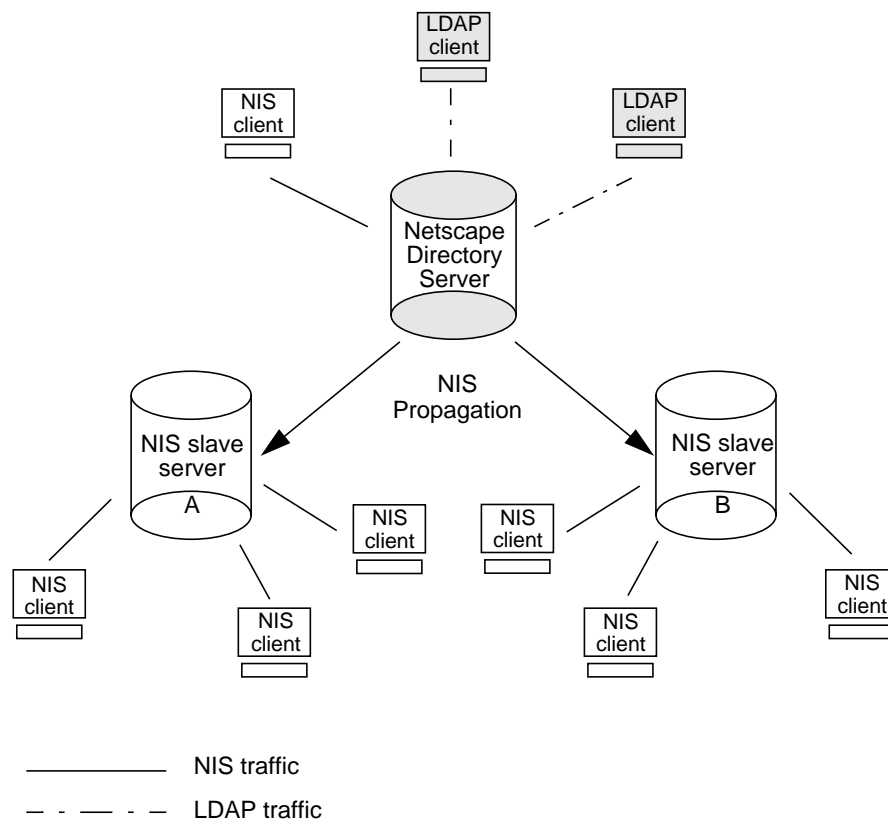


Figure 1.2 Mixed NIS-Netscape Directory Server Environment

Figure 1.2 shows the first stage in the transition to a Netscape Directory Server environment synchronized with NIS. The recommended approach is to replace a master server first so that you can perform write operations on the NIS information in the LDAP directory. You can replace a slave server first, but you will only be able to perform read operations on the NIS information in the LDAP directory.

At this stage, you have the choice between two methods for performing updates to NIS information:

- Using the standard NIS procedure (modifying the `/etc` files)
- Making modifications in the LDAP database

In both cases, it is the standard NIS propagation process, `yppush` that handles the replication of data to slave servers. The difference is that if you choose the second method for updates, the NIS source files on the master server will no longer be up-to-date.

Note Do not use a combination of both methods.

For more information on performing updates to NIS information, refer to “Updating NIS Maps” on page 33.

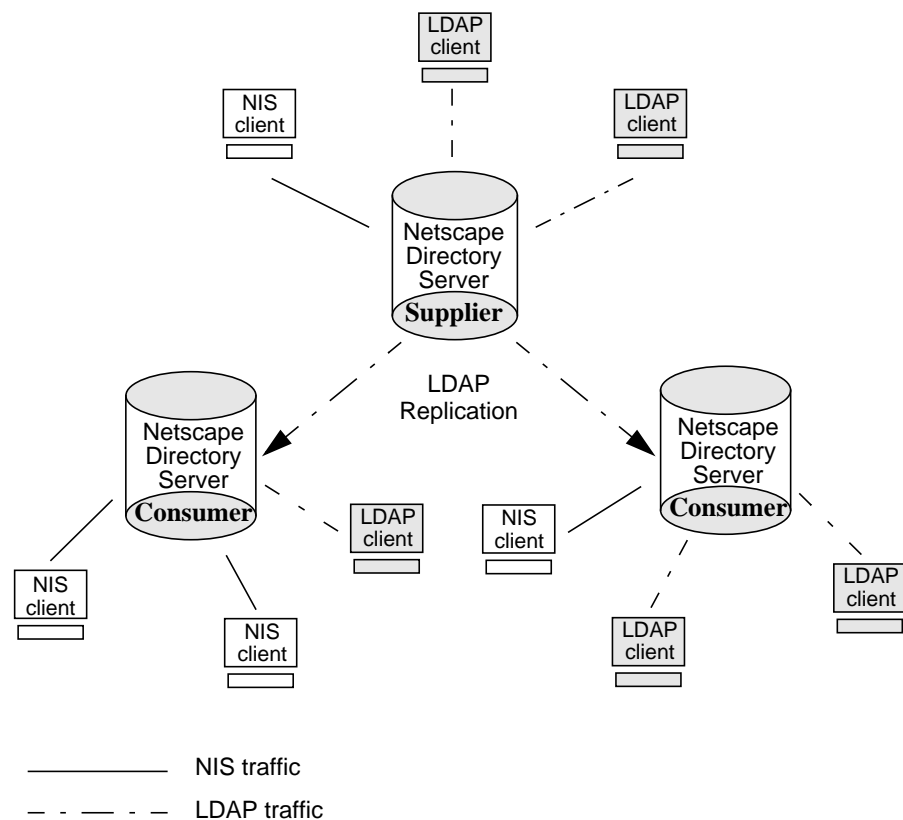


Figure 1.3 Netscape Directory Server Environment

Figure 1.3 shows an environment where all standard NIS servers are synchronized with the Netscape Directory Server. All your servers support both NIS and LDAP clients.

At this stage too, you have the choice of methods for performing updates to NIS information:

- Using the standard NIS update procedure (modifying the `/etc` files)
- Making modifications in the LDAP database

You also have the choice of replication methods:

- Using the standard NIS propagation process, `yppush`
- Using LDAP replication

The second method, using LDAP replication is the most efficient.

Daemons Providing Synchronized Service

When you synchronize your standard NIS service with the Netscape Directory Server, some of the standard NIS daemons are stopped and replaced by Netscape Directory Server daemons that are functionally equivalent. Table 1.1 shows how all the functions in the standard NIS environment are preserved, and the daemons that provide them.

This section also provides a brief description of the additional commands and daemons provided with Solaris Extensions for Netscape Directory Server 4.11. Full reference information on these and on the standard NIS processes are provided in Chapter 5, “NIS Command & File Reference.”

Standard NIS	Netscape Directory Server	Functional Purpose
<code>ypserv</code>	<code>dsypserv</code>	Server process that responds to NIS requests
<code>rpc.yppasswdd</code>	<code>dsyppasswdd</code>	Process that modifies the NIS <code>passwd</code> table
<code>ypxfrd</code>	-	Process that synchronizes NIS master and slave servers
<code>yppush</code>	-	Command initiated on the master server to propagate updates to slave servers
<code>ypxfr</code>	<code>dsypxfr</code>	Command initiated on an NIS slave server to request updates from the NIS master server
<code>makedbm</code>	<code>dsmakedbm</code>	Process that builds the NIS tables from the NIS administration files
<code>ypinit</code>	<code>dsypinit</code>	Process that initializes NIS processes on NIS clients and NIS servers
<code>rpc.nisd_resolv</code>	-	Process that provides DNS access
-	<code>dsypsync</code>	Process that resynchronizes specified NIS tables with the NIS information stored in the LDAP directory.

Table 1.1 Daemons and Commands Providing NIS Service

dsypserv

The `dsypserv` daemon provides the NIS service. It receives and responds to NIS requests in the same way as the `ypserv(1M)` daemon. It does not convert NIS requests to LDAP requests and NIS responses to LDAP responses. It uses all of the standard NIS administration and lookup utilities such as `yppoll(1M)`, `yppush(1M)`, `ypcat(1)`, `ypwhich(1)`, and `ypmatch(1)`. The major difference between `dsypserv` and `ypserv` is that `dsypserv` uses the information in the LDAP directory to build the NIS maps.

Some of the standard NIS utilities were modified to accommodate the synchronization with the LDAP directory. These are listed in Table 1.1.

For more detailed information on the NIS service, refer to the man page for `ypserv(1M)`.

dsyppasswdd

The `dsyppasswdd` daemon manages changes to the password tables stored in the LDAP directory. It runs on the master server and responds to requests from users who invoke the `passwd` command to change their password, full name (in the `gecos` field) or shell. The `dsyppasswdd` daemon makes updates to the LDAP database, and also to the NIS `passwd` file. It also updates the `shadow` file when there is one.

For details of how to configure the `dsyppasswdd` daemon, refer to “`dsyppasswdd`” on page 126.

dsypxfr

The `dsypxfr` command is used on a slave server to request updates from the master. It can be invoked manually or by a `crontab` file, or called by the `dsypserv` daemon.

For more information on `dsypxfr`, refer to “`dsypxfr`” on page 129.

dsmakedbm

The `dsmakedbm` command builds the NIS tables from the information held in the NIS source files. It is slightly different from the standard NIS `makedbm` command because it calls the `dsimport` utility to create NIS entries in the directory.

dsypinit

The `dsypinit` command initializes an NIS server as a master server or slave server. This command can also be used instead of the NIS standard `ypinit` command to initialize NIS clients.

For details of how to use `dsypinit`, refer to “`dsypinit`” on page 122.

Note Do not run `dsypinit` to initialize the NIS service with Netscape Directory Server. Use `dsypinstall`, as described in “Initializing Synchronized Operation” on page 25.

dsypsync

The `dsypsync` command can be used to resynchronize NIS tables with the information held in the LDAP directory. In this synchronization process, the reference data, is the data held in the LDAP directory.

Note The `dsypsync` command regenerates the NIS maps, however, it does not update the NIS source files.

For more information on `dsypsync`, see “`dsypinit`” on page 122.

dsyp

The `dsyp` command can be used to initialize or disable the NIS/LDAP synchronized service. It is called by the `dsypinstall` script to initialize or stop the NIS synchronization with the LDAP directory. It can also be invoked manually.

For more information on `dsyp`, refer to “`dsyp`” on page 118.

dsypaddmap

The `dsypaddmap` commands simplifies the process of creating NIS-to-LDAP mapping definitions. You can use `dsypaddmap` to automatically create an NIS-to-LDAP mapping for a new map you want to import into the directory.

Note Although `dsypaddmap` creates a new subtree in the directory for the new map, it does not import NIS data into the directory. You must use `dsimport` to create the NIS entries in the directory.

For more information on `dsypaddmap`, refer to “`dsypaddmap`” on page 119.

dsypdelmap

The `dsypdelmap` command disables NIS maps. It does not actually remove the NIS entries from the directory because they could be used by other applications. It does not remove the mapping definition either.

For more information on `dsypdelmap`, refer to “`dsypdelmap`” on page 120.

Daemons Providing Synchronized Service

Initializing and Operating the NIS Service

This chapter explains how to initialize the NIS extension of the Netscape Directory Server. The NIS service provided by the Solaris Extensions for Netscape Directory Server 4.11 preserves your existing NIS environment. It takes into account the customizations that you may have made to the standard NIS environment.

This chapter includes the following sections:

- “Initializing Synchronized Operation” on page 25
- “Access Control on NIS Information” on page 30
- “Configuring the NIS Service” on page 31
- “Updating NIS Maps” on page 33
- “Propagating NIS Maps” on page 34
- “Adding and Deleting NIS Maps” on page 35

Initializing Synchronized Operation

This section explains the steps you need to follow to set up synchronized NIS/LDAP operation. Because this requires that you have a running NIS server, this section also explains the main steps in setting up a standard NIS server.

Setting Up an NIS Server

You must set up an NIS server if you plan to install Netscape Directory Server and the Solaris Extensions on a machine that has not previously been used as an NIS server.

If you are migrating a current NIS server to synchronized operation with LDAP, go directly to “Setting up Synchronized Service” on page 26.

To set up an NIS server, follow these steps:

1. Copy the NIS administration files from the NIS master server in your network to your new server. Their default location is under the `/etc` directory.
2. Copy the Makefile from the NIS master server in your network to your new server. Its default location is `/var/yp`.
3. Install and initialize the Solaris NIS server. See the man page for `ypinit(1M)`.

If you need detailed instructions on how to set up an NIS server, refer to the documentation for your Solaris operating environment.

Setting up Synchronized Service

To migrate an existing NIS server to synchronize it with the Directory Server, you must:

1. Verify that the NIS server daemon `ypserv` is running.
2. Back up your current NIS files and database.
3. Install and configure Netscape Directory Server 4.11. For information, refer to the Netscape Directory Server product documentation.
4. Install Solaris Extensions for Netscape Directory Server 4.11. For information, refer to *Solaris Extensions Installation Guide*.
5. Run the `dsypinstall` script, as described in “Running the dsypinstall Script” on page 27.
6. Make sure the NIS administrator has access to the NIS entries in the directory.

The initialization script, `dsypinstall`, modifies your NIS Makefile so that the make process calls `dsmakedbm` and `dsimport` instead of the standard NIS `makedbm` command. It also stops the `ypserv` daemon and starts the `dsypserv` daemon instead to fulfill the same functions.

The `dsmakedbm` command creates the NIS binary tables on disk in the same way as the `makedbm` command. In addition, it creates LDAP entries in the directory from the information contained in the NIS source files. For information on the location of NIS entries in your directory tree, see “NIS Files/LDAP Subtrees” on page 71.

Running the `dsypinstall` Script

When the package installation is complete, a message indicates that you must run the `dsypinstall` script to initialize the NIS service. This script will prompt you to provide all the configuration information required to start operating the NIS service.

You will need to understand and prepare the information you must supply to the `dsypinstall` script. The script prompts you for:

1. The name of the NIS domain managed by the server.
The name you provide is used to create the directory subtree under which all NIS entries are stored.
2. The installation directory for the Netscape Directory Server.
3. The DN of the Netscape Directory Server directory manager.
The DN you provide must be the same as the one you provided in the setup script for the Netscape Directory Server. This DN has all permissions on the Netscape Directory Server. By default, it is `cn=Directory Manager`.
4. The port number where the directory server listens for LDAP traffic.
5. The DN of the administrator for NIS information.
You can use the DN of the directory manager or create a special entry for the NIS administrator. You must make sure the NIS administrator is granted all permissions on the NIS subtrees in the directory. Refer to “Access Control on NIS Information” on page 30.
6. The location of the NIS source files.

The `dsypinstall` script assumes that your Makefile is located in `/var/yp`. It also assumes that the source files for NIS tables are all located in the directory that you specify when prompted, except for the `aliases` file which is assumed to be in `/etc/mail`.

7. A list of NIS servers.

When you have all this information, you are ready to run `dsypinstall`:

1. Back up your current NIS files and database.
2. As root, run `dsypinstall`:

```
# /opt/SUNWconn/sbin/dsypinstall
```

When the `dsypinstall` script has successfully finished, the NIS server is initialized and the LDAP directory database contains the information extracted from the NIS tables.

For details on how NIS information is imported and stored in the LDAP directory, refer to Chapter 4, “NIS Information in the LDAP Directory”.

Verifying your NIS Installation

This section explains some of the methods you can use to ensure that NIS information is present in the directory, and that the NIS service is running according to the information you provided in the initialization script.

Checking NIS Information

There are several ways of checking that NIS information is present in the directory:

- By using Deja to browse the NIS subtrees. For information on starting and using Deja, refer to Chapter 3, “Using Deja to Update NIS Information.”
- By using the `ldapsearch` utility.

For example, if you want to get a list of hosts for the `airius.com` domain, you can use the following command:

```
% ldapsearch -D "cn=Directory Manager" -w passwd -b  
"ou=Hosts,ou=Services,dc=airius,dc=com" objectclass=ipHost cn=*
```

This command returns a list of the `ipHost` type entries in the directory.

All NIS maps except the aliases map, are stored under the `ou=Services` subtree in the directory. For more information on the location of NIS information in the LDAP directory tree, refer to “NIS Files/LDAP Subtrees” on page 71.

Checking Server Status

If you want to check the functional role of a server (master or slave), you must look in the `/etc/opt/SUNWconn/ldap/current` directory. You will see either a `nis.master` or a `nis.slave` file which indicates the role fulfilled by the NIS server.

Getting the List of Supported Maps

There are several ways of getting the list of NIS maps supported by a server:

- By using the `ldapsearch` utility
- By using the `ypwhich` command

Using `ldapsearch`, to get a list of the NIS maps supported in the `airius.com` domain, you can use the following example command:

```
% ldapsearch -D "cn=Directory Manager" -w passwd -b
"ou=admin,ou=Services,dc=airius,dc=com" objectclass=sunnismap
sunnismapfullname=*
```

The `ou=admin` subtree holds information that is used internally by the NIS synchronization processes. For more information on `ldapsearch`, refer to the *Netscape Directory Server Administrator's Guide*.

To get a list of NIS maps using the NIS command, `ypwhich`, type:

```
% ypwhich -m
```

This command returns a list of the NIS maps followed by the name of the server that masters them.

Access Control on NIS Information

So that the NIS service can operate, the NIS information stored in the directory needs to be accessible to the NIS administrator you defined in the `dsypinstall` process. When the `dsypinstall` process has finished, there aren't any access control rules defined in the directory for the NIS subtrees. Defining access to the information is dependent on whether:

- You defined the NIS administrator to be the same as the directory manager for the Netscape Directory Server

If you defined the NIS administrator to be the same as the directory manager for the Netscape Directory Server by providing the same DN and password in the `dsypinstall` script, then the NIS administrator automatically has access to the NIS subtrees in the directory. You do not need to create a special access control instruction (ACI). This also means that you have just one directory manager for all the information stored in the directory.

- You defined the NIS administrator to be different from the directory manager for the Netscape Directory Server (recommended option)

If you defined the NIS administrator to be different from the directory manager for the Netscape Directory Server, you must create an ACI in the directory server to grant access to the NIS administrator. You must create an ACI of the form:

```
aci: <NIS_subtree> <attributes> <permissions> <nis_admin_DN> *
```

where:

- *NIS_subtree* is the subtree you specified in `dsypinstall` to store NIS entries
- *attributes* is the list of attributes to which the ACI applies
- *permissions* specifies the permissions granted on the subtree and attributes
- *nis_admin_DN* is the DN you specified in `dsypinstall` for the directory manager

For example, the default ACI for the `dc=airius,dc=com` subtree is:

```
aci:
(target="ldap:///dc=airius,dc=com")
(targetattr=*) (version 3.0; aci "NIS managing"; allow ( all ) userdn =
"ldap:///cn=nis-admin,dc=airius,dc=com" ;)
```

For full details on creating ACIs, refer to the *Netscape Directory Server Administrator's Guide*.

Configuring the NIS Service

When you first initialize the NIS service using `dsypinstall`, you are prompted to provide values for all the NIS configurations parameters. You can change this configuration at any time after the initial configuration by editing the `nis.mapping` file.

For your changes to be taken into account, after modifying the appropriate parameters, you must run the `dsypinit` script. On a master server, run the following command:

```
# dsypinit -m
```

On a slave server, run the following command:

```
# dsypinit -s <master>
```

where *master* specifies the name of the master server which will replicate data to the slave.

These commands will modify the LDAP directory database to take into account the configuration changes you made in the `nis.mapping` file.

Setting NIS Configuration Variables

All configuration information is stored in the first part of the `nis.mapping` file under a section entitled *Configuration Variables*.

DOMAIN_NAME

Specifies the NIS domain managed by the server.

NAMING_CONTEXT

When this variable is defined, it specifies the directory tree suffix (or naming context) under which the NIS subtree is created.

If this variable is not defined, the directory tree suffix is derived from the domain name supplied when running the `dsypinstall` script as described in “Running the `dsypinstall` Script” on page 27. By default, the directory tree suffix is

generated with dc (domain component) attributes. For example, with `DOMAIN_NAME=france.airius.com`, the directory tree suffix created by default is `dc=france,dc=airius,dc=com`.

The NIS subtree shown in “NIS Files/LDAP Subtrees” on page 71 is created under this subtree.

`ADMIN_SUFFIX`

The distinguished name of the subtree that will hold NIS administrative entries. These entries are maintained automatically by the server.

`DBM_DIRECTORY`

Specifies the directory where the NIS binary maps are generated.

`AUTOMATIC_PUSH`

When NIS entries are modified in the LDAP directory, specifies to automatically push modifications to slave NIS servers. This variable is used only in the context of standard NIS replication (using `yppush`), not in the context of LDAP replication.

The possible values for this variable are **enabled** or **disabled**. The default setting is **disabled**.

`AUTOMATIC_PUSH_DELAY`

Specifies the delay for pushing modifications to slaves in minutes. When this variable is defined, the `AUTOMATIC_PUSH` variable must be **enabled**.

Configuring NIS Subtrees

The subtrees created for NIS entries during the initialization of the NIS service are specified in the `nis.mapping` file by the keyword `BASE_DN`. This base DN is the concatenation of an organizational unit (ou) specific to each map, and of a `rootTree` token that is usually common to several maps.

For example, the subtree for the entries created from the `/etc/networks` file is defined by the following two lines in the `nis.mapping` file:

```
rootTreeT=ou=Services,$NAMING_CONTEXT | |ou=Services,$DC_NAMING
BASE_DN=ou=Networks,$rootTreeT
```

The directory entries created from the `/etc/networks` file are created under the `ou=Networks,ou=Services` subtree.

The choice of a naming structure through the `NAMING_CONTEXT` keyword or `DC_NAMING` keyword is a configuration decision.

The `DC_NAMING` keyword contains a domain component (dc) suffix. The DNs of entries created with that naming structure have a suffix of the form `dc=sun, dc=com`. This is the default choice when you initialize the NIS service, because the import process derives a dc naming suffix from the domain name you supply when you run `dsypinstall`.

If you prefer to use a different naming structure, you must un-comment the `NAMING_CONTEXT` keyword at the beginning of the `nis.mapping` file, under the Common section for the front-end. Change the value of the `NAMING_CONTEXT` keyword to specify the suffix under which you want NIS entries to be created. The value you specify must be a valid suffix or subtree in the directory tree held on the directory server.

After changing the suffix, you must run the `dsypinit` script. For information, refer to “`dsypinit`” on page 122.

Note Do not comment out the `DOMAIN_NAME` keyword in the `nis.mapping` file. This keyword contains the domain name that you supplied during the `dsypinstall` process.

Updating NIS Maps

Once you have populated the directory for the first time, you have two options for data maintenance:

- Make updates to the entries in the directory using an LDAP client, for example the Netscape Console, or Deja

Making updates to the entries in the directory is the most efficient method of maintaining NIS information. The NIS maps are automatically updated when changes are made to NIS information in the directory. However, the NIS source files are *not* updated, therefore they will become obsolete. If you want to resynchronize the NIS source files with the directory content, you can export the NIS entries to the corresponding NIS files by using `dsexport`. For details, see “`dsexport`” on page 110.

- Make updates to the NIS source files, and run `make` in the `/var/yp` directory

When you make updates to the NIS source files, the NIS information in the LDAP directory is automatically updated. However, in some cases, such as when you rebuild the LDAP database using the `ldif2db` command, or if you experience a system crash, you might find that the NIS information in the directory is no longer in sync with the NIS source files. In such cases, you need to resynchronize them using the `dsimport` utility. For information on `dsimport`, refer to “`dsimport`” on page 114.

Propagating NIS Maps

There are two methods of propagating NIS maps between master servers and slave servers. Between two Netscape Directory Servers, choose LDAP replication. Between a Netscape Directory Server and a legacy NIS server, you must use standard NIS replication.

Do not use both LDAP replication and standard NIS replication on the same subtrees or individual entries. As a general rule, use only one replication method between two servers.

Standard NIS Replication

If you make updates to your NIS files rather than to NIS entries in the directory, when you run `make` to rebuild the NIS tables, the `yppush` command is automatically executed.

LDAP Replication

If you make updates to NIS entries in the directory, you use LDAP replication to push changes to consumer NIS servers. For information on configuring LDAP replication, see *Netscape Directory Server Administrator's Guide*.

Adding and Deleting NIS Maps

After you have run `dsypinstall` to initialize the synchronized NIS/LDAP naming service, you can still modify the server configuration to add support for new NIS maps or remove maps that are no longer used. These operations are performed using the `dsypaddmap` and `dsypdelmap` commands respectively. For information, see “`dsypaddmap`” on page 119, and “`dsypdelmap`” on page 120.

These operations can be performed without stopping and restarting the NIS server.

Adding and Deleting NIS Maps

Using Deja to Update NIS Information

This chapter explains how to use the Deja tool to add, delete and modify NIS information in the LDAP directory.

The Netscape Directory Server provides several graphical interfaces to view or modify information in the directory:

- The Directory Console
- The Directory Express web gateway
- Deja

The Directory Console can be used to create and modify most information in the directory but it does not offer dedicated templates for creating and modifying NIS information.

The Directory Express web gateway is designed for viewing the contents of the directory quickly, searching for entries, and modifying some directory information. Its limited functionality makes it unsuitable for more complex operations.

Deja is a Java™ directory editor particularly suited for the day-to-day management of NIS and RADIUS information. With the tool you can search for and view entries, create and modify entries, delete entries, and copy and paste entries. Deja can be connected remotely or locally to a Netscape Directory Server.

This chapter includes the following sections:

- “Introduction to Deja” on page 38

- “General Operations” on page 42
- “Operations on NIS Entries” on page 46
- “Setting Deja Properties” on page 59

Introduction to Deja

Deja provides a comprehensive user interface suitable for maintaining the directory contents. Figure 3.1 shows the Deja Create panel. The tool is split into four areas, the toolbar, the browser window, the function window, and the status bar. The toolbar, browser window, and status bar can be hidden








When you click on an icon in the toolbar or select an option from the Directory menu, the appropriate screen is displayed in the function window.







Figure 3.1 Deja Directory Editor

The toolbar offers quick access to the most commonly used functions. Refer to Table 3.1 for a description of the icons and their functions.

Table 3.1 Deja Toolbar Icons

Icon	Function
Login	Click this icon to login to the directory server. You must login to modify the contents of the directory.
	
Search	Click this icon to search for entries in the directory.
	
View	Select an entry in the directory browser window and click this icon to view the entry's attributes and values.
	
Create	Click this icon to create a new entry in the directory.
	
Modify	Select an entry in the directory browser window and click this icon to modify the entry's properties.
	
Rename	Select an entry in the directory browser window and click this icon to modify the Relative Distinguished Name of the entry.
	
Delete	Select an entry in the directory browser window and click this icon to remove an entry from the directory.
	

Icon	Function
Cut 	Select an entry in the directory browser window and click this icon to cut the entry from the directory, and retain a copy in the clipboard.
Copy 	Select an entry in the directory browser window and click this icon to copy the entry into the clipboard.
Paste 	After an entry has been cut or copied to the clipboard, select a parent entry in the directory browser window and click this icon to paste the entry as a child of the selected entry
Help 	Gives the URL to follow to display the online help.

Starting Deja

Deja must connect to the directory server. This connection can be established only if the `ns-slapd` daemon is running on the directory server. If the `ns-slapd` daemon is not running, Deja will start but is unable to connect.

For information on starting the Netscape Directory Server, see the *Netscape Directory Server Administrator's Guide*.

To display Deja:

1. Run `dejasync` on the server. As `root` type:

```
# /opt/SUNWconn/ldap/sbin/dejasync
```

For details on the options of the `dejasync` command, refer to “`dejasync`” on page 108. You must run `dejasync` to make sure that Deja takes into account all the configuration options you set during the `dsypinstall` process.

2. On the machine running the directory server daemon, `ns-slapd`, set the `JAVA_HOME` environment variable to the installation directory of your Java Virtual Machine (JVM).

3. Type:

```
prompt% /opt/SUNWconn/bin/deja [ hostname [:port_number]]
```

where:

- *hostname* is the hostname of the directory server. The default is `localhost`.
- *port_number* is the port number of the directory server. The default is 389.

Note The machine on which you are running Deja needs to have a Java Virtual Machine and JDK version 1.1.5 or a compatible version installed.

Logging In

Directory access rights are defined by a set of access control rules on the directory server. You must be the directory administrator to modify the access control rules. When you log in to the directory, your username and password are compared with those stored in the directory. If there is a match, the access rights defined in the access control rules are granted.

You can browse the directory content without logging in, but you must have write permission before you can modify directory entries. Figure 3.2 shows the Login panel.

Note It may not be possible to browse the directory content without logging in. This depends on the access control rules defined in the directory server.

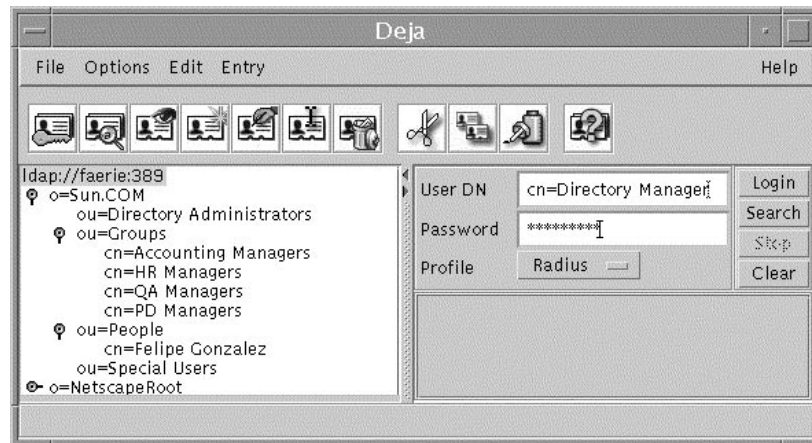


Figure 3.2 Deja Login Panel

To log in to Deja:

1. Click on the Login icon or select Login from the File menu.
2. Type the Distinguished Name (DN) of the NIS administrator in the User text field.
You can define an alias for the NIS administrator in `Deja.properties` file.
See “Setting Deja Properties” on page 59 for information on creating a login alias.
3. Type your password in the Password field.
4. Select the profile (Standard, NIS or RADIUS) you want from the Profile option button.
The default profile is Standard.
5. Click Login.
Your password is compared to the password stored in the directory. If there is no match the login fails.

General Operations

This section gives some tips on how to use Deja.

Setting the Display Options

The Options menu is used to hide or show the toolbar, status bar, or directory browser. The default view has all of these elements.

To hide or show an element, select it from the Options menu to change its status.

Setting Deja Properties

The Deja Properties panel displays information about the selected user profile, and the connection to the directory server. To access the Properties panel, select Properties from the File menu.

The Properties panel is displayed, and shows the user properties and connection properties of Deja. See Figure 3.3.



Figure 3.3 Deja Properties Panel

User Properties

The User Properties pane displays the name of the connected user and the user profile for creating or modifying entries.

Name

If you are not logged into the directory server, Anonymous is displayed. If you have logged in, the login name is displayed.

User Profile

To set the user profile, select the profile (Standard, NIS or RADIUS) from the Profile option button in the User Properties pane.

The default profile is Standard.

Connection Properties

The Connection Properties pane displays the name of the directory server to which Deja is connected, and the connection port number.

Server and Port Number

Deja displays information about its connection to the directory server. The default port number that Deja uses to connect to the directory server is 389. The host name and port number can be specified on the command line when Deja is started. See “Starting Deja” on page 40.

To connect to a different directory server or change the port number from within Deja see “Connecting to Another Directory Server” on page 45.

Opening a New Deja Window

To open a new window in Deja, from the File menu select New Window. The new window has its own connection to the directory server. This means that you can connect to several directory servers simultaneously.

Closing a Deja Window

To close a Deja window, select Close from the File menu. The Deja window is closed.

To close all Deja windows, select Exit from the File menu. A confirmation window is displayed. Click Yes to close all Deja windows.

Reconnecting Deja to the Directory Server

If the directory server is disabled for some reason, Deja loses its connection to the directory. Deja does not automatically reconnect to the directory server when it is re-enabled.

To reconnect Deja to the directory server, select Connect from the File menu. Deja is reconnected.

Connecting to Another Directory Server

1. To connect Deja to a different directory server, select Connect To... from the File menu.

The Connect To... dialogue box is displayed.

2. Deja tries to connect to the new directory server. If it is unable to connect, an error message is displayed.

Refreshing the Browser Window

If directory operations are being performed on the same directory server by another user or by the administrator, the browser window is not automatically updated. To refresh the browser window:

1. In the browser window, click on the root entry of the branch you want to refresh.
You can choose to refresh all of the directory by selecting the directory root entry, or to refresh just a branch by clicking on the root entry of the branch.
2. From the File menu, select Refresh Subtree.

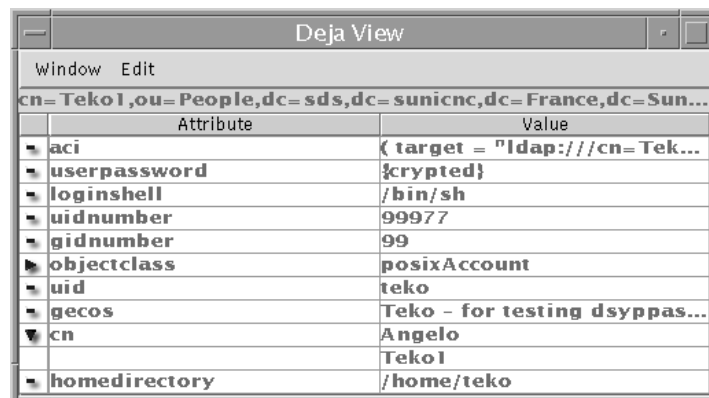
All the branches of the directory below the selected entry are collapsed in the browser window. When they are reopened, they are refreshed.

Operations on NIS Entries

This chapter describes the read, create, modify, delete and search operations that can be performed on directory entries using Deja. Deja offers specific templates for searching for and creating NIS entries. To view the NIS-specific search and create panels you must change the Deja user profile to NIS, as explained in “User Properties” on page 43.

Viewing an Entry

Use View to look at the attributes defined for an entry in the directory. Figure 3.4 shows the Deja View window. You can only open one View window per entry. To refresh a View window after modifying an entry, view the entry again. The original View window is replaced with a new one.



Deja View	
Window Edit	
cn= Teko1,ou= People,dc= sds,dc= sunicnc,dc= France,dc= Sun...	
Attribute	Value
aci	{ target = "ldap:///cn= Teko1,ou= People,dc= sds,dc= sunicnc,dc= France,dc= Sun..." }
userpassword	{crypted}
loginshell	/bin/sh
uidnumber	99977
gidnumber	99
objectclass	posixAccount
uid	teko
gecos	Teko - for testing dsypas...
cn	Angelo
	Teko1
homedirectory	/home/teko

Figure 3.4 Deja View Window

When an attribute has more than one value, an arrow is displayed next to the attribute name in the entry definition: a right arrow when the values are collapsed, and a down arrow when the values are expanded.

The View Window

There are three ways to display the View window:

- Double-click on an entry in the browser window
- Select an entry in the browser window and click on the View icon, or from the Entry menu, select View
- Double click on an entry in the search results list

Closing a View Window

To close a View window, select Close from the Window menu of the View window. Alternatively, you can double click on the Window menu button.

Copying an Entry From a View Window

To copy an entry from a View window, select Copy from the Edit menu of the View Window. The entry is copied to the clipboard.

Highlighting an Entry From a View Window

To highlight an entry in Deja's browser window from the View Window, select Highlight from the Edit menu.

Creating a New Entry

The Deja create panel offers templates that guide you through the creation of an NIS entry. The available templates are Users, Aliases, Hosts, and Groups. These templates respectively represent entries for the passwd, aliases, hosts, and groups maps. When the NIS maps are imported into the LDAP directory, an LDAP subtree is created for each map.

Table 3.2 shows the default templates available in Deja, the corresponding NIS map, and the corresponding subtree in the LDAP directory.

Table 3.2 Deja Templates

Template Name	Map Name	LDAP Subtree (default)
Users	/etc/passwd	ou=People,o=Sun,c=US
Aliases	/etc/mail/aliases	ou=Aliases,o=Sun,c=US
Hosts	/etc/hosts	ou=Hosts,o=Sun,c=US
Groups	/etc/groups	ou=Group,o=Sun,c=US

To modify the default NIS templates, or create new ones, you can modify the `Deja.properties` file on the directory server. See “Setting Deja Properties” on page 59 for information.

Figure 3.5 shows the Deja Create panel for NIS users.

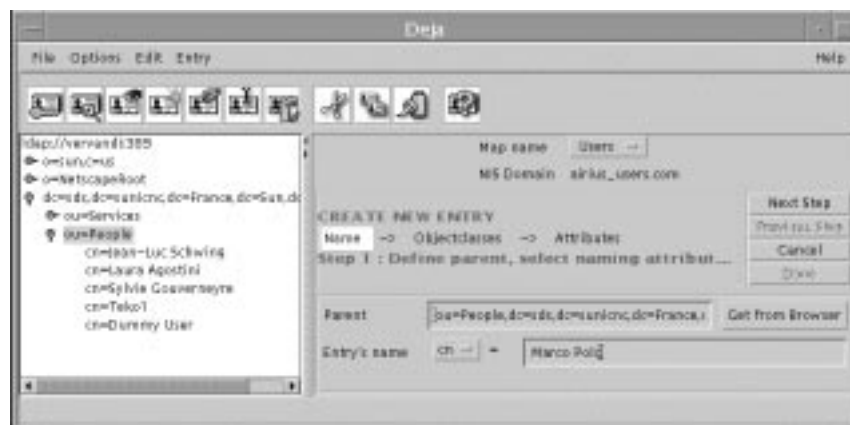


Figure 3.5 Deja Create Panel for NIS Users

1. Click on the Create icon or select Create from the Entry menu.

The Create panel is displayed.

There are three steps to creating an NIS directory entry. You must complete each step before you can progress to the next one. Click on Next Step and Previous Step to navigate between the steps.

- Name the entry. See “Naming an Entry” on page 49.
- Select object classes for the new entry. See “Selecting Object Classes” on page 50.

- Add values to the mandatory and optional attributes for each object class. See “Selecting Attributes” on page 50.
2. When you have completed the entry definition, click Done.

Naming an Entry

1. Provide the DN of the parent entry. There are several ways to achieve this:
 - Select the name of the map to which the new entry should be added from the Map Name option button. This automatically adds the default parent name to the Parent text field. The default parent name depends on the selected map, and is defined in the `Deja.properties` file on the directory server.
 - In the browser window, click once on the parent to select it, then click the Get From Browser button next to the Parent text field. The Distinguished Name of the selected entry is imported to the Parent text field.
2. Name the entry by selecting a naming attribute with the option button next to the Entry’s name field.
 The list of naming attributes is defined in the `Deja.properties` file on the directory server.
3. Type the Relative Distinguished Name of the entry in the Entry Name text field.

When you are satisfied with the entry name and parent, click the Next Step button to select object classes and attributes.

See “Selecting Object Classes” on page 50 for information on selecting object classes. The Select Object Classes window is displayed pre-filled with default object classes depending on the selected template. The default object classes are specified in the `Deja.properties` file on the directory server.

See “Selecting Attributes” on page 50 for information on selecting attributes for the entry. The attributes available for selection are defined for each object class in the schema.

Selecting Object Classes

You can define one or more object classes for your entry. When the objectclass list is complete, click the Next Step button to select attributes. If you have selected the Users map in the NIS profile, the required object classes are already listed in the Selected Objectclasses pane.

Note If the selected object classes do not contain the previously selected naming attribute, a warning message is displayed. You must either specify a different naming attribute by going back to the first step, or add an appropriate object class to the entry.

Adding an Object Class to the Entry

To add an object class to the entry, double click on the object class from the Available Objectclasses list.

Alternatively, you can select an object class from the Available Objectclasses list and click on the right arrow button to add the object class to the entry.

Removing an Object Class From the Entry

To remove an object class from an entry, double click on the object class in the Selected Objectclasses list.

Alternatively, you can select the object class in the Selected Objectclasses list and click on the left arrow button to remove the object class from the entry.

Selecting Attributes

Each object class has a number of mandatory and optional attributes associated with it. An entry definition table, with the current list of attributes and values is displayed in the right pane. Mandatory attributes are marked with (M), optional attributes with (O).

The names of the mandatory attributes are already listed in the entry definition before you assign a value to them. To complete the entry, you *must* provide values for these attributes. If you try to add an entry to the directory without assigning values to all the mandatory attributes, an error message is displayed.

For example, if you want to create an entry with the person and posixAccount object classes, the mandatory attributes are:

- last name (sn, or surname)

- login name (uid, or userid)
- user id number (uidNumber)
- group id number (gidNumber)
- home directory (homeDirectory)

Optional attributes for these object classes can include description, see also, telephone number and userPassword.

Some attributes accept multiple values, others can only have one value. By default, attributes are multi-valued. Single-valued attributes are identified in the schema by the `SINGLE-VALUE` keyword. If you try to add more than one value to a single-valued attribute, an error message is displayed.

Assigning a Value to an Attribute

To assign a value to an attribute:

1. From the Choose Attribute list, or from the entry definition, select the attribute for which you want to add a value.

For example, select Login Name from the Choose Attribute list.

2. Type the value for the attribute in the text field.

For example, type **mpolo** in the text field.

3. Click Add to add the value of the attribute to the entry definition.

The value appears in the entry definition next to the attribute. See Figure 3.6.

4. To add an additional value for an attribute, repeat steps 1 to 3.

You must add values for all the mandatory attributes displayed in the entry definition table.

5. Double click on the entry in the browser to display all of its attributes.

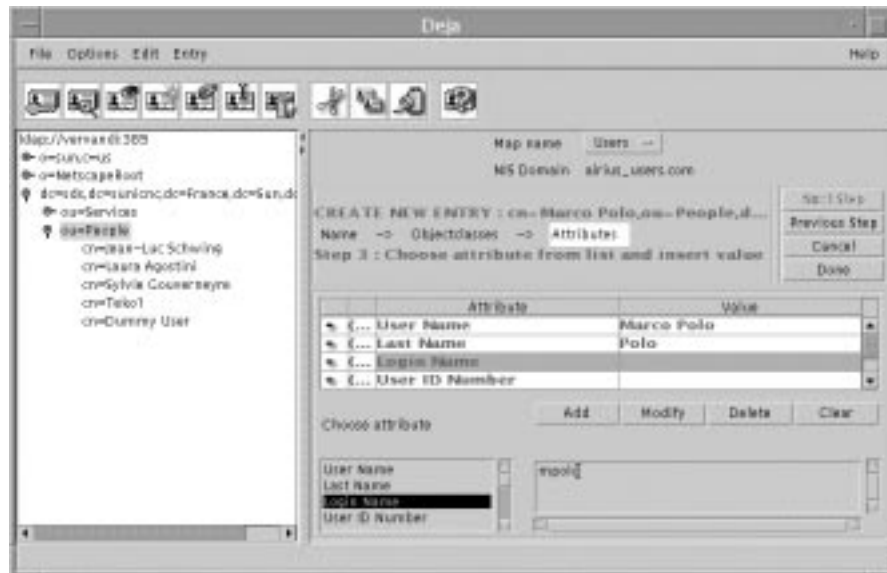


Figure 3.6 Example Entry Create Window

Deleting a Value From an Attribute

To delete an attribute value:

1. Select the value or the attribute name in the entry definition.
2. Click Delete.
 - If you delete the only value for an optional attribute, the attribute is removed from the entry definition.
 - If you delete the only value for a mandatory attribute, the value is cleared from the entry definition. The attribute stays in the definition.

Modifying an Attribute Value

To modify an attribute value:

1. Select the value of the attribute you want to modify in the entry definition.

The attribute value appears in the text field.

2. Modify the value and click Modify.

The modified value appears in the entry definition.

Cancel

To cancel a create operation at any time, in the Create panel, click Cancel. The entry definition is cleared.

Deleting an Entry

The delete panel of Deja is used to delete entries from the directory. Figure 3.7 shows the Deja Delete panel.



Figure 3.7 Deja Delete Panel

You must have write permission for the entry you want to delete. See “Logging In” on page 41 for information.

1. Select the entry you want to delete in the browser window.

You can only delete leaf entries. You cannot delete a root entry such as the root DSe or a parent that still has children.

2. Click on the Delete icon, or select Delete from the Entry menu.

The Delete panel is displayed.

3. Click on Delete to remove the entry from the directory.
4. Click on Cancel to clear the delete panel.

Warning There is no undelete function.

Cut, Copy and Paste

This section explains how to perform cut, copy and paste operations on directory entries using Deja.

Cutting an Entry

Use Cut to remove an entry from the directory and keep a copy of it on the clipboard. The entry can be pasted from the clipboard into the directory in another location.

You must have write permission for the entry you want to cut. See “Logging In” on page 41 for information.

To cut an entry from the directory:

1. In the browser, click on the entry you want to cut.
2. Click on the Cut icon. Alternatively, select Cut from the Edit menu, or press Ctrl-x on the keyboard.

The entry is cut from the directory to the clipboard. You can now paste the entry to a new location in the directory.

3. If you want to restore the entry to the directory, select Restore from the Edit menu.
The entry is restored to its original position in the directory, if possible. If the parent entry no longer exists, or has been renamed, the paste is not possible and an error message is displayed.

Copying an Entry

Use Copy to copy an existing entry from the directory into the clipboard. The entry can then be pasted from the clipboard into the directory in another location.

To copy an entry in the directory:

1. In the browser, click on the entry you want to copy to select it.
2. Click on the Copy icon. Alternatively, select Copy from the Edit menu, or press Ctrl-c on the keyboard.

The entry is copied from the directory to the clipboard.

You can now paste the entry to a new location in the directory.

Pasting an Entry

After a Cut or Copy operation, use Paste to paste an entry from the clipboard into the directory. You can paste at different levels in the directory tree:

- At the same level as the entry you copied or cut into the clipboard
- At any level in the directory tree

You must have write permission to paste an entry into the directory. See “Logging In” on page 41 for information.

1. To copy an entry and paste it at the same level in the subtree:

Immediately following the copy operation, click on the Paste icon. Alternatively, select Paste from the Edit menu, or press Ctrl-v on the keyboard.

In the browser window, the pasted entry is displayed. A sequence number is appended to its name to ensure naming remains unique at a given level in the directory tree.

2. To cut or copy an entry and paste it at a different level:

Select the new parent entry for the entry you want to paste, and click on the Paste icon. Alternatively, select Paste from the Edit menu, or press Ctrl-v on the keyboard.

To copy an entry, and paste it immediately below the copied entry, you must click elsewhere in the directory tree to deselect the copied entry, then click on it again to select it, then perform the paste. If you do not deselect then reselect, the entry in the clipboard is pasted at the same level, not one level below.

Restoring an Entry

If you accidentally cut an entry from the directory, you can restore it, provided that you have not performed any subsequent cut or copy operations.

To restore an entry that you have just cut from the directory, select Restore from the Edit menu. The entry on the clipboard is returned to its original location.

Modifying an Entry

Use Modify to change attributes and object classes in directory entries. The Deja Modify panel is very similar to the attribute selection panel that you use to create an entry. See Figure 3.5.

You must have write permission for the entry that you want to modify. See “Logging In” on page 41 for information.

1. In the browser, click on the entry you want to modify.
2. Click on the Modify icon or select Modify from the Entry menu.
The Modify Attributes window is displayed. Click on the Modify Objectclasses button to display the Modify Objectclasses window.
3. You can modify the following characteristics of an entry:
 - The values for the mandatory and optional attributes for each object class. See “Selecting Attributes” on page 50.
 - The object classes for the entry. See “Selecting Object Classes” on page 50.
 - If you want to change the name of the entry, use the Rename function. See “Renaming an Entry” on page 56.
4. When you have finished the modifications, click Done.

Renaming an Entry

Use Rename to modify the Relative Distinguished Name (RDN) of an entry. Figure 3.8 shows the Deja Rename panel.

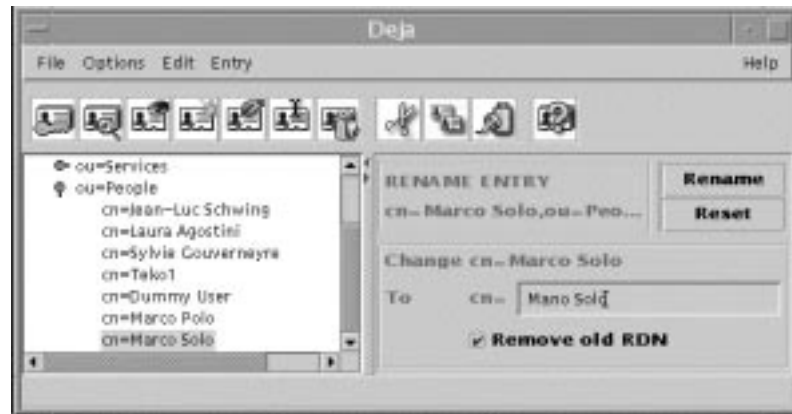


Figure 3.8 Deja Rename Panel

You must have write permission for the entry you want to rename. See “Logging In” on page 41 for information.

1. Select the entry you want to rename in the browser window.
You can only rename leaf entries. You cannot rename parents that still have children, or the root entry.
2. Click on the Rename icon, or select Rename from the Entry menu.
The rename panel appears. The name of the parent and the Relative Distinguished Name (RDN) of the selected entry are displayed.
3. Type the new RDN of the entry in the To text field.
If you want the new RDN to replace the old RDN, check the Remove old RDN check box.
By default the new RDN replaces the old RDN. If the Remove old RDN check box is unchecked, the new RDN is added to the entry as an additional value.
4. Click the Rename button.

Searching for an Entry

Use Search when you want to find an NIS entry in the directory. By default, you can search four NIS maps with this function; users, aliases, hosts and groups. Figure 3.9 shows the Deja Search panel for NIS users.

The `Deja.properties` file on the directory server defines the NIS search templates available in Deja. For information on adding or modifying NIS templates see “Setting Deja Properties” on page 59.



Figure 3.9 Deja Search Panel for NIS Users

To search for an NIS entry:

1. Click on the Search icon, or select Search from the Entry menu.
The Search panel is displayed.
2. Select the map you want to search from the Map Name option button.
Default options are Users, Aliases, Hosts or Groups.
3. Type the text string you want to search for in the NIS Key text field.
The search can include the wildcard character *.
4. Click Search to start the search.

The search results are displayed in the search results list and the number of entries found is displayed in the status bar. If there are no matches, the search results list is empty and the status bar indicates that no entries were found.

5. To stop the search at any time, click the Stop button.
The search is stopped and no results are returned.
6. Click the Clear button to clear the search text field.

Search Results List

Search results are displayed in a list below the search criteria.

The headings of the search results table depend on the map that is searched. Table 3.3 shows the attribute types displayed in the search result table for each map.

Table 3.3 NIS Search Results Lists

Map Name	Attributes
users	cn, uid, uidNumber, gidNumber, homeDirectory
aliases	cn, rfc822mailMember
hosts	cn, ipHostNumber, macaddress
groups	cn, gidNumber, memberUid

To view an entry from the search results list, double-click on the entry's name. The view entry window is displayed, and the entry is highlighted in the browser window.

Setting Deja Properties

This section describes how to configure Deja properties, and the maintenance operations required to synchronize Deja properties with configuration changes that occur on the directory server side.

Many of Deja's characteristics can be configured by the directory administrator. The characteristics are defined in the `Deja.properties` file on the directory server.

File Structure

The `Deja.properties` file is located in the `/opt/SUNWconn/ldap/html` directory on the directory server.

The `Deja.properties` file consists of four sections:

- “General Properties” on page 61
- “Standard LDAP Properties” on page 62
- “NIS Properties” on page 64
- RADIUS properties

Some of the properties described in the `Deja.properties` file are not relevant to the topics discussed in this book. In particular, this section does not explain how to create new search filters, or the meaning of the RADIUS parameters.

File Syntax

Each section in the `Deja.properties` file contains a list of definitions. Each definition ends with a carriage return. The different elements in a definition are separated by commas. Related elements are separated by semi-colons.

For example, the list of supported NIS maps is defined as follows:

```
NIS_MAPS=passwd.byname;NIS_MAP_USERS_CHOICE,  
mail.aliases;NIS_MAP_ALIASES_CHOICE, hosts.byname;NIS_MAP_HOSTS_CHOICE,  
group.byname;NIS_MAP_GROUPS_CHOICE
```

In this example, each map name and the label identifying it are separated by semi-colons. The four map elements in the definition are separated by commas. This definition does not show the actual labels that appear in Deja’s menus. These are defined separately, in the *localized resource bundle*. The localized resource bundle contains translations in every supported locale for the user interface of Deja.

Labels

Standard Deja labels and identifiers (parameters ending in `_LABEL`, `_IDENTIFIER` or `_CHOICE`) are defined in the localized resource bundle. You cannot change these definitions. You can, however, create your own labels.

For example, if you want to add support for the networks NIS map, you must append the map name, and a new label to identify it to the NIS maps definition. In the following example, the `networks.byaddr` map is identified by the label `Networks`.

```
NIS_MAPS=passwd.byname;NIS_MAP_USERS_CHOICE,  
mail.aliases;NIS_MAP_ALIASES_CHOICE, hosts.byname;NIS_MAP_HOSTS_CHOICE,  
group.byname;NIS_MAP_GROUPS_CHOICE, networks.byaddr;Networks
```

This definition is local to your `Deja.properties` file. It is not part of the localized resource bundle.

General Properties

In the General Properties section the following parameters are defined:

`SCHEMA_THREAD_TIME_LIMIT`

Defines a time limit in milliseconds on the time it takes Deja to read the schema. The default value is no time limit.

`BROWSER_ENTRY_LIMIT`

Specifies the maximum number of entries that can be displayed in the browser. If a limit has been set, you must refresh certain subtrees before opening more. The default value is no limit.

`BROWSER_SUBENTRY_LIMIT`

Defines the maximum number of immediate children of an entry that can be displayed in the browser. The default value is no limit.

`BROWSER_LOAD_SUBNODES_TIME_LIMIT`

Specifies the maximum amount of time allowed for Deja to load the children of a node when the node is opened in the browser. This is not the amount of time it then takes to display those children. The default value is 10000 milliseconds.

`BROWSER_CHECK_NODE_TIME_LIMIT`

This is the maximum time taken for Deja to verify whether an entry is a leaf or a node. The default value is 2000 milliseconds.

`STANDARD_SECURITY_AUTHENTICATION`

Defines the standard authentication mechanism used in the login panel. The only possible value for this parameter is **simple**.

The following example shows the General Properties section of the `Deja.properties` file.

```
# schema thread time limit in milliseconds (0 = no limit)
SCHEMA_THREAD_TIME_LIMIT=0
```

Setting Deja Properties

```
#
# manage referrals as entries (true or false)
REFERRALS_MANAGE_DSA=true
#
# max. number of nodes in browser tree (0 = no limit)
BROWSER_ENTRY_LIMIT=0
# max number of subnodes of a node in the browser tree (0 = no limit)
BROWSER_SUBENTRY_LIMIT=0
# time limit to load subnodes (in ms, 0 = no limit)
BROWSER_LOAD_SUBNODES_TIME_LIMIT=10000
# time limit to verify if entry is a leaf or an inner node (in ms, 0 =
no limit)
BROWSER_CHECK_NODE_TIME_LIMIT=2000
#
# authentication mechanism
# supported values : CRAM-MD5, simple (cleartext password)
STANDARD_SECURITY_AUTHENTICATION=simple
# STANDARD_SECURITY_AUTHENTICATION=CRAM-MD5
```

Standard LDAP Properties

In the Standard LDAP Properties section of the `Deja.properties` file you can:

- Specify which attribute values are hidden in Deja
- Specify parameters for the login panel
- Define new standard search filters

This section describes the first two items.

Hiding Attributes

`STANDARD_ATTRIBUTES_CRYPTED`

In the View, Modify and Create windows of Deja, some attribute values are not displayed, or replaced by a localized text string. You can specify the attributes you want to be hidden by adding them to the `STANDARD_ATTRIBUTES_CRYPTED` list. Attribute names are separated by commas. By default the values for `userpassword`, `radiusppppasswd`, `radiusloginpasswd`, `chappassword`, and `radiusslippasswd` are hidden.

Login Parameters

STANDARD_LOGIN_SEARCH_FILTER

The search feature of the login panel operates using the filter defined with this label. By default it is `(| (cn=*{0}*) (uid=*{0}*))`. This search filter means that either the cn attribute or the uid attribute should contain the search string typed by the user in the search text field.

STANDARD_LOGIN_MAX_SEARCH_RESULT

Specifies the maximum number of search results per naming context returned by a login search. The default value is 55.

STANDARD_LOGIN_ALIASES

Defines an alias for the user DN you use to login to Deja. By default, there are no aliases defined, and the `STANDARD_LOGIN_ALIASES` parameter is commented out. The definition in the `Deja.properties` file reads as follows:

```
# STANDARD_LOGIN_ALIASES= userA_alias; userA_dn; userB_alias; userB_dn
```

To add a login alias, you must uncomment the line, add an alias name and a user DN for login. For example, if the user `cn=Robert Travis, ou=sales, o=sun, c=us` wants to login frequently, you can create an alias for him, for example, `rob`. To add this alias, you would edit the `STANDARD_LOGIN_ALIASES` definition in the `Deja.properties` file to read as follows:

```
STANDARD_LOGIN_ALIASES= rob; cn=Robert Travis,ou=sales,o=sun,c=us
```

Note If you create several aliases, you must use a semi-colon to separate them, and not a comma, which is the standard syntax, because the comma is used to separate the different elements in the DN. The semi-colon separates the elements of a DN from a new alias definition.

For example, if you also wanted to add an alias for an administrator user whose DN is `cn=Directory Manager, o=sun, c=us`, the `STANDARD_LOGIN_ALIASES` definition in the `Deja.properties` file would read as follows:

```
STANDARD_LOGIN_ALIASES= rob ; cn=Robert Travis,ou=sales,o=sun,c=us ;  
Directory Manager ; cn=Directory Manager, o=sun, c=us
```

When Deja is restarted the aliases are available in the Login panel. This parameter is case-sensitive.

NIS Properties

The NIS Properties section in the `Deja.properties` file contains a number of tokens defining how NIS entries are displayed in Deja.

The `NIS_MAPS` definition contains a list of the NIS maps displayed in Deja. Each map listed in `NIS_MAPS` is associated with the following tokens: `NIS_FILTER`, `NIS_DOMAIN`, `NIS_ROOT`, `NIS_NAMINGATTR`, and `NIS_OCLASS`. The values for these tokens are copied from the `nis.mapping` file by the `dejasync` utility. If the tokens already exist in the `Deja.properties` file they are updated. For more information, see “`dejasync`” on page 108.

When you create new map definitions in the `nis.mapping` file, these must also be declared in the `Deja.properties`. This procedure is described in “Adding a NIS Map to Deja using `dejasync`” on page 66.

`NIS_MAPS`

Specifies the list of maps available in Deja. Each map name is followed by a semicolon and the label that appears in the Map Name option button of the NIS Search, Create or Modify panels. If you create a new map in the `nis.mapping` file, you must declare the map name in the `NIS_MAPS` token in the `Deja.properties` file. The syntax is:

```
NIS_MAPS= map.name;map_label, map.name;map_label, ...
```

`NIS_FILTER.map.name`

Specifies the filter that is used in the NIS Search panel. This definition is automatically generated by running `dejasync`.

`NIS_DOMAIN.map.name`

Specifies the label that appears in the NIS Create, Modify and Search panels. It shows to which domain the NIS map applies. This definition is automatically generated by running `dejasync`.

`NIS_NAMINGATTR.map.name`

Specifies the naming attributes that are available in the NIS Create panel. This is a comma separated list. This definition is automatically generated by running `dejasync`.

NIS_ROOT.map.name

Specifies the DN of the root entry used for NIS searches. It is also the default parent entry displayed in the NIS Create panel. This definition is automatically generated by running `dejasync`.

NIS_OCLASS.map.name

Specifies the default object classes that are added to an entry definition in the NIS Create Panel. This is a comma separated list. This definition is automatically generated by running `dejasync`.

NIS_LIST.map.name

Contains names of the attributes and header labels for the NIS search results table. The syntax is:

```
NIS_LIST.map.name= attribute;header_label, attribute;header_label, ...
```

NIS_ADD.map.name

Specifies labels and syntax for attributes in the NIS Create panel. The syntax is:

```
NIS_ADD.map.name= attribute;label:syntax, attribute;label:syntax, ...
```

Where syntax is one of the four basic input types (**int**, **string**, **crypt** and **ipaddr**). If a syntax isn't specified, the default value, **string**, is used. Specifying a syntax is useful to constrain user input:

- **crypt** is a text field where each character typed in is replaced with *.
- **string** is a text field that accepts any character.
- **int** is a text field that accepts only integer numbers.
- **ipaddr** consists of four **int** fields, in the format *int.int.int.int*.

NIS_LIST.default

Contains the names of the attributes listed in the NIS search results table if **NIS_LIST** is not defined for a map.

Adding a NIS Map to Deja using dejasync

To add a NIS map definition to Deja from the `nis.mapping` file using `dejasync`:

1. Login as `root` or become `superuser` on the directory server.
2. Add the name of the map, and the label you want to be used in the NIS Maps option button, to the `NIS_MAPS` definition.

The map must have a mapping definition in the `nis.mapping` file. For example, to add the `ethers.byname` map to Deja using the label `Ethers`, the `NIS_MAPS` definition will look like this:

```
NIS_MAPS= ethers.byname;Ethers,
passwd.byname;NIS_MAP_USERS_CHOICE,
mail.aliases;NIS_MAP_ALIASES_CHOICE,
hosts.byname;NIS_MAP_HOSTS_CHOICE,
group.byname;NIS_MAP_GROUPS_CHOICE
```

3. Run `dejasync` by typing:

```
prompt# /opt/SUNWconn/ldap/sbin/dejasync -d Deja_properties_directory
-n NIS_mapping_file
```

Where:

- *Deja_properties_directory* specifies the directory containing the `Deja.properties` file. By default, this is `/opt/SUNWconn/ldap/html`.
- *NIS_mapping_file* specifies the filename of the NIS mapping file. By default, this is `/etc/opt/SUNWconn/ldap/current/mapping/nis.mapping`.

The `dejasync` utility reads the map declarations in `NIS_MAPS`, reads the mapping definitions from the `nis.mapping` file, and updates or adds the corresponding map definitions to the `Deja.properties` file.

For the example, the following map definition is added at the end of the `Deja.properties` file:

```
NIS_OCLASS.ethers.byname= ieee802Device
NIS_FILTER.ethers.byname=
(& (objectClass=ieee802Device) (cn=$NIS_KEY))
NIS_NAMINGATTR.ethers.byname=cn
```

```
NIS_ROOT.ethers.byname=dc=airius,dc=com
NIS_DOMAIN.ethers.byname=airius.com
```

4. Optionally add NIS_LIST and NIS_ADD definitions for the new map.
5. Exit from Deja and restart it to use the new map.

For the example, the following definitions are added for NIS_LIST and NIS_ADD:

```
NIS_LIST.ethers.byname=cn;Host Name, macAddress;Ethernet Address,
description;Comments

NIS_ADD.ethers.byname=cn;Host Name, macAddress;Ethernet Address,
description;Comments
```

Default NIS Map Definitions

In the `Deja.properties` file, the section containing the NIS properties is as follows:

```
# list of supported maps

#
NIS_MAPS=passwd.byname;NIS_MAP_USERS_CHOICE,
mail.aliases;NIS_MAP_ALIASES_CHOICE, hosts.byname;NIS_MAP_HOSTS_CHOICE,
group.byname;NIS_MAP_GROUPS_CHOICE

#
# passwd map
#
NIS_FILTER.passwd.byname= (&(objectclass=posixAccount)(uid=$NIS_KEY))
NIS_DOMAIN.passwd.byname= airius_users.com
NIS_NAMINGATTR.passwd.byname=cn
NIS_ROOT.passwd.byname= dc=airius_users,dc=com
NIS_OCLASS.passwd.byname= posixaccount, person
NIS_LIST.passwd.byname=cn;NIS_USER_CN_ATTR_LABEL,
uid;NIS_UID_ATTR_LABEL, uidNumber;NIS_UIDNUMBER_ATTR_LABEL,
gidNumber;NIS_GIDNUMBER_ATTR_LABEL,
homeDirectory;NIS_HOMEDIRECTORY_ATTR_LABEL
NIS_ADD.passwd.byname=cn;NIS_USER_CN_ATTR_LABEL, sn;NIS_SN_ATTR_LABEL,
uid;NIS_UID_ATTR_LABEL, uidNumber;NIS_UIDNUMBER_ATTR_LABEL;int,
gidNumber;NIS_GIDNUMBER_ATTR_LABEL;int,
homeDirectory;NIS_HOMEDIRECTORY_ATTR_LABEL,
```

Setting Deja Properties

```
userPassword;NIS_USERPASSWORD_ATTR_LABEL;crypt,
loginShell;NIS_LOGINSHELL_ATTR_LABEL,
description;NIS_DESCRIPTION_ATTR_LABEL

#

# alias map

#
NIS_FILTER.mail.aliases= (&(objectclass=nisMailAlias)(cn=$NIS_KEY))
NIS_DOMAIN.mail.aliases= airius_aliases.com
NIS_NAMINGATTR.mail.aliases=cn
NIS_ROOT.mail.aliases= dc=airius_aliases,dc=com
NIS_OCLASS.mail.aliases= nismailalias
NIS_LIST.mail.aliases= cn;NIS_ALIAS_CN_ATTR_LABEL,
rfc822mailMember;NIS_RFC822MAILMEMBER_ATTR_LABEL
NIS_ADD.mail.aliases= cn;NIS_ALIAS_CN_ATTR_LABEL,
rfc822mailMember;NIS_RFC822MAILMEMBER_ATTR_LABEL

#

# host map

#
NIS_FILTER.hosts.byname= (&(objectclass=ipHost)(cn=$NIS_KEY))
NIS_DOMAIN.hosts.byname= airius_hosts.com
NIS_NAMINGATTR.hosts.byname=cn
NIS_ROOT.hosts.byname= dc=airius_hosts,dc=com
NIS_OCLASS.hosts.byname= ipHost
NIS_LIST.hosts.byname= cn;NIS_HOST_CN_ATTR_LABEL,
ipHostNumber;NIS_IPHOSTNUMBER_ATTR_LABEL,
macaddress;NIS_MACADDRESS_ATTR_LABEL
NIS_ADD.hosts.byname= cn;NIS_HOST_CN_ATTR_LABEL,
ipHostNumber;NIS_IPHOSTNUMBER_ATTR_LABEL;ipaddr,
macaddress;NIS_MACADDRESS_ATTR_LABEL, 1;NIS_L_ATTR_LABEL

#

# group map

#
NIS_FILTER.group.byname= (&(objectclass=posixGroup)(cn=$NIS_KEY))
NIS_DOMAIN.group.byname= airius_groups.com
NIS_NAMINGATTR.group.byname=cn
```

```

NIS_ROOT.group.byname=    dc=airius_groups,dc=com
NIS_OCLASS.group.byname=  posixGroup
NIS_LIST.group.byname=    cn;NIS_GROUP_CN_ATTR_LABEL,
gidNumber;NIS_GIDNUMBER_ATTR_LABEL, memberUid;NIS_MEMBERUID_ATTR_LABEL
NIS_ADD.group.byname=     cn;NIS_GROUP_CN_ATTR_LABEL,
gidNumber;NIS_GIDNUMBER_ATTR_LABEL;int,
memberUid;NIS_MEMBERUID_ATTR_LABEL

NIS_LIST.default=        cn;NIS_CN_ATTR_LABEL

```

Setting Deja Properties

NIS Information in the LDAP Directory

This chapter shows the organization of NIS information in the LDAP directory tree. Solaris Extensions for Netscape Directory Server 4.11 offers a default tree structure for NIS information that is compliant with RFC 2307 *An Approach for Using LDAP as a Network Information Service*. However, this structure can be configured to suit your own preferences.

This chapter also explains how the information in NIS maps is extracted and used to create LDAP entries.

This chapter includes the following sections:

- “NIS Files/LDAP Subtrees” on page 71
- “NIS File Entries/LDAP Entries” on page 72
- “NIS Schema” on page 82
- “NIS/LDAP Mapping Summary” on page 97

NIS Files/LDAP Subtrees

During the NIS initialization phase, the information imported into the LDAP directory is organized according to the tree structure shown in Figure 4.1.

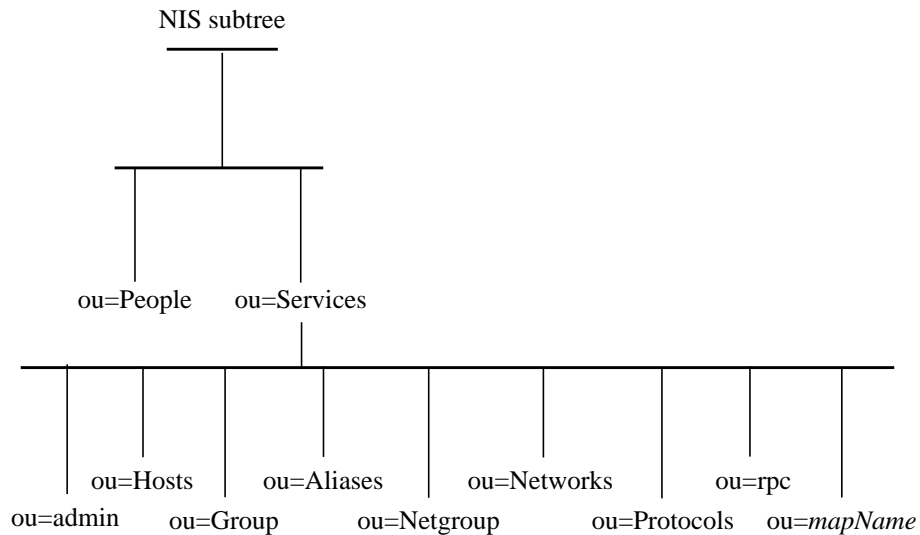


Figure 4.1 NIS Subtrees in the DIT

In this figure, *mapName* represents any map not defined in the default `nis.mapping` file, for example `services.byname`.

This tree structure can be configured to suit your needs by modifying the mapping file `/etc/opt/SUNWconn/ldap/current/mapping/nis.mapping`.

NIS File Entries/LDAP Entries

The NIS information imported into the LDAP directory is extracted from standard NIS files and from your custom-built files. Solaris Extensions for Netscape Directory Server 4.11 provides a mapping of standard NIS entries to LDAP attributes. The default `nis.mapping` file explicitly describes the information mapping for the following NIS files:

- `aliases`, see “aliases Mapping” on page 74
- `bootparams`, see “bootparams Mapping” on page 76
- `ethers`, see “ethers Mapping” on page 77

- `group`, see “group Mapping” on page 78
- `hosts`, see “hosts Mapping” on page 78
- `netgroup`, see “netgroup Mapping” on page 79
- `networks`, see “networks Mapping” on page 80
- `passwd`, see “passwd Mapping” on page 80
- `protocols`, see “protocols Mapping” on page 81
- `rpc`, see “rpc Mapping” on page 81
- `ypservers`, see “ypservers Mapping” on page 82

The mapping syntax for the `aliases` file is commented in detail in “aliases Mapping” on page 74. The description of all other maps is limited to the mapping of attributes. For full details on mapping syntax and semantics, refer to Appendix A, “Mapping Syntax and Semantics.”

The `nis.mapping` file also contains a generic mapping definition for files that are not specifically defined. This includes any custom-built maps listed in your Makefile, and the following standard NIS files:

- `auto_home`
- `auto_master`
- `netid`
- `netmasks`
- `publickey`
- `services`
- `timezone`

A mapping definition for each of these files and any custom-built files is added automatically in the `nis.mapping` file when you run `dsypinstall`. These mapping definitions are created from the generic mapping described in “Generic Mapping” on page 74 and are identical for every map, except for the map name.

The `nis.mapping` file is used by the following components:

- The `dsimport` utility, to extract information from NIS files and convert it to LDAP directory entries.
- The `dsexport` utility, to extract information from LDAP entries and restore the NIS files in their original format.
- The `dejasync` command, to synchronize the `nis.mapping` file with the `deja.properties` file. For information of the `deja.properties` file, refer to “Deja.properties” on page 99.

- The `dsypaddmap` and `dsypdelmap` utilities to respectively add an NIS map in the directory, or to delete an NIS map from the directory.

The mapping definitions in the `nis.mapping` file specify the following:

- The object class and attributes used to create an entry in the directory
- The subtree under which the entries are created

Generic Mapping

A generic mapping is applied to all files listed in the NIS `Makefile` that do not have a specific mapping definition in the `nis.mapping` file. This mapping extracts the NIS key and NIS value information from any file and creates an LDAP entry as follows:

LDAP attribute	NIS information
<code>cn</code>	<code>nisKey</code> (case ignored in LDAP searches)
<code>sunNisKey</code>	<code>nisKey</code>
<code>nisMapEntry</code>	<code>nisValue</code>
<code>nisMapName</code>	map name in <code>Makefile</code>
<code>objectClass</code>	<code>top</code> , <code>nisSunObject</code>

aliases Mapping

An entry in the `aliases` file is usually of the form:

```
aliasname: mailaddress1, mailaddress2, mailaddress3...
```

The mapping of this line in the `nis.mapping` file is shown below:

```
Table: mail.aliases
...
Import:
    Extract:
        LINE =>$aliasNameT:$aliasListT
    Condense:
        rfc822mailMembersT=string2instances($aliasListT, ",")
        trimrfc822mailMembersT=trim($rfc822mailMembersT)
        objectClassT=string2instances("top nisMailAlias", " ")
    Build:
        dn=cn=$aliasNameT,$BASE_DN
```

```
cn=$aliasNameT
rfc822mailMember=$trimrfc822mailMembersT
objectClass=$objectClassT
```

The object class and attributes necessary to create an entry are listed in the Import/Build section of the mapping definition. The values assigned to the LDAP attributes listed in the Build section are taken from the tokens specified in either:

- The Import/Extract section of the mapping definition
- The Import/Condense section of the mapping definition
- The Common section in the `nis.mapping` file that applies to all mapping definitions.

Extract Section

In the Extract section, the keyword `LINE` shows the initial decomposition of a line in the `aliases` file. If, in your `aliases` file you use a punctuation mark other than a colon to separate the alias name from the alias list, you must change the `LINE` definition to match your file format.

Condense Section

In the Condense section, the token definition for `rfc822mailMembersT` provides a further decomposition of the alias list. If in your `aliases` file you use a punctuation mark other than a comma to separate the elements in the alias list, you must change the token definition to match your file format.

The token definition for `trimrfc822mailMemberT` removes any unnecessary white spaces from the result produced by the preceding `string2instances` operation on the alias list. The `string2instances` and `trim` functions are described in Appendix A, “Mapping Syntax and Semantics.”

The token definition for `objectClassT` shows the `nisMailAlias` object class, and the object classes it inherits from, that is the top object class.

Build Section

In the Build section, the DN of the entry is created by concatenating the `cn` attribute and attribute value with the `BASE_DN` token. The `BASE_DN` token identifies the subtree under which the entry is created.

The `rfc822mailMember` and `objectClass` attributes are multi-valued. There is one occurrence of these attributes for each value resulting from the `string2instances` operations.

Example

For example, the `aliases` file in the domain `France.airius.com` contains the following line:

```
dir-team: pdurand, jdupond, asantini, msmith, dphilippe, smartin
```

The directory entry created from this line in the `aliases` file is:

```
dn: cn=dir-team, ou=Aliases, ou=Services, dc=France, dc=airius, dc=com
cn: dir-team
rfc822mailMember: pdurand
rfc822mailMember: jdupond
rfc822mailMember: asantini
rfc822mailMember: msmith
rfc822mailMember: dphilippe
rfc822mailMember: smartin
objectclass: nisMailAlias
objectclass: top
```

bootparams Mapping

An entry in the `bootparams` file is usually of the form:

```
hostname parameter1 parameter2 parameter3...parameterx
```

For example, the `bootparams` file in the domain `France.airius.com` contains the following line:

```
camembert    root=server1:/export/camembert/root \
              swap=server1:/export/camembert/swap \
              domain=France.airius.com
```

The directory entry created from this entry in the `bootparams` file is:

```
dn: cn=camembert, ou=Hosts, ou=Services, dc=France, dc=airius, dc=com
cn: camembert
bootParameter: root=server1:/export/camembert/root
bootParameter: swap=server1:/export/camembert/swap
bootParameter: domain=France.airius.com
objectClass: top
objectClass: device
```

```
objectClass: bootableDevice
```

The host `camembert` probably also has an entry in `/etc/ethers` and in `/etc/hosts`. However, in the LDAP directory, the host `camembert` has just one entry, with all the attributes derived from the `ethers` mapping and the `hosts` mapping. The LDAP entry created for `camembert` has several object classes:

- One inherited structural object class: `device`
- Three auxiliary object classes; `bootableDevice`, `ieee802Device`, and `ipHost`

Based on the examples given in “ethers Mapping” on page 77, and in “hosts Mapping” on page 78, the complete entry created in the LDAP directory for host `camembert` is:

```
dn: cn=camembert, ou=Hosts, ou=Services, dc=France, dc=airius, dc=com
cn: camembert
cn: bertie
bootParameter: root=server1:/export/camembert/root
bootParameter: swap=server1:/export/camembert/swap
bootParameter: domain=France.airius.com
macAddress: 0:1:23:aa:bb:cc
ipHostNumber: 123.456.789.1
description: SS5 Pierre's desktop
objectClass: top
objectClass: device
objectClass: bootableDevice
objectClass: ieee802Device
objectClass: ipHost
```

If the entry for `camembert` is deleted from the `bootparams` file, the directory entry for `camembert` is updated by removing the `bootableDevice` object class and the `bootParameter` attributes which are specific to that object class.

ethers Mapping

An entry in the `ethers` file is usually of the form:

```
ethernetaddress      machinename
```

For example, the `ethers` file in the domain `France.airius.com` contains the following line:

```
0:1:23:aa:bb:cc  camembert
```

The directory entry created from this line in the `ethers` file is:

```
dn: cn=camembert, ou=Hosts, ou=Services, dc=France, dc=airius, dc=com
cn: camembert
cn: bertie
macAddress: 0:1:23:aa:bb:cc
objectClass: top
objectClass: device
objectClass: ieee802Device
```

The host `camembert` may also have an entry in the `etc/bootparams` file and the `/etc/hosts` file. See the example given in “bootparams Mapping” on page 76 for details of the complete LDAP directory entry created from these files.

group Mapping

An entry in the `group` file is usually of the form:

```
groupname:password:groupidnumber:listofmembers
```

For example, the `group` file in the domain `France.airius.com` contains the following line:

```
sysadmin:yai957KJwXrjc:10:pdurand, jdupond, asantini
```

The directory entry created from this line in the `group` file is:

```
dn: cn=sysadmin, ou=Group, ou=Services, dc=France, dc=airius, dc=com
cn: sysadmin
gidNumber: 10
memberUid: pdurand
memberUid: jdupond
memberUid: asantini
userPassword: {crypt}yai957KJwXrjc
```

hosts Mapping

An entry in the `hosts` file is usually of the form:

```
ipaddress hostname hostaliasnames #hostdescription
```

For example, the `hosts` file in the domain `France.airius.com` contains the following line:

```
123.456.789.1    camembert    bertie    # SS5 Pierre's Desktop
```

The directory entry created from this line in the `hosts` file is:

```
dn: cn=camembert, ou=Hosts, ou=Services, dc=France, dc=airius, dc=com
cn: camembert
cn: bertie
ipHostNumber: 123.456.789.1
description: SS5 Pierre's desktop
objectClass: top
objectClass: device
objectClass: ipHost
```

The host `camembert` may also have an entry in the `/etc/bootparams` file and the `/etc/ethers` file. See the example given in “bootparams Mapping” on page 76 for details of the complete LDAP directory entry created from these files.

netgroup Mapping

An entry in the `netgroup` file is usually of the form:

```
netgroupname grouptriple grouptriple grouptriple grouptriple
```

where *grouptriple* is of the form (*hostname, username, domainname*).

For example, the `netgroup` file in the domain `France.airius.com` contains the following line:

```
printers\
(bordeaux,-,France.airius.com)\
(bourgogne,-,France.airius.com)\
(sauternes,-,France.airius.com)
```

The directory entry created from this entry in the `netgroup` file is:

```
dn: cn=printers, ou=netgroup, ou=Services, dc=France, dc=airius, dc=com
cn: printers
nisNetGroupTriple: bordeaux,-,France.airius.com
nisNetGroupTriple: bourgogne,-,France.airius.com
nisNetGroupTriple: sauternes,-,France.airius.com
```

networks Mapping

An entry in the `networks` file is usually of the form:

```
networkname networkaddress networkalias #description
```

For example, the `networks` file in the domain `France.airius.com` contains the following line:

```
airius-eng 123.456.789 eng #engineering subnetwork
```

The directory entry created from this line in the `networks` file is:

```
dn: cn=airius-eng, ou=networks, ou=Services, dc=France, dc=airius,
dc=com
cn: airius-eng
cn: eng
ipNetworkNumber: 123.456.789
description: engineering subnetwork
objectClass: top
objectClass: ipNetwork
```

passwd Mapping

An entry in the `passwd` file is usually of the form:

```
userid:userPasswd:uidnumber:gidnumber:gecos:homeDir:shell
```

When there is a shadow file associated with the `passwd` file, it is usually of the form:

```
userid:userPasswd:::::::
```

The user ID in the `passwd` file and in the shadow file are identical, but the user's password is actually stored in the shadow file and not in the `passwd` file.

For example, the `passwd` file in the domain `France.airius.com` contains the following line for Pierre Durand:

```
pdurand:x:12345:67:Pierre Durand-Project Manager:/home/pdurand:/bin/csh
```

The `x` instead of the user password indicates that the actual password is stored in the shadow file. The shadow file contains the following line for Pierre Durand:

```
pdurand:yai957KJwXrjc:::::::
```


The directory entry created from this line in the `passwd` file is:

```
dn: uid=pdurand, ou=People, dc=France, dc=airius, dc=com
cn: Pierre Durand
uid: pdurand
userPassword: {crypt}yai957KJwXrjc
uidNumber: 12345
gidNumber: 67
gecos: Pierre Durand-Project Manager
homeDirectory: /home/pdurand
loginShell: /bin/csh
objectClass: top
objectClass: account
objectClass: posixAccount
```

protocols Mapping

An entry in the `protocols` file is usually of the form:

```
protocolname protocolnumber protocolalias #description
```

For example, the `protocols` file in the domain `France.airius.com` contains the following line:

```
tcp    0    TCP_Protocol    #Transmission Control Protocol
```

The directory entry created from this line in the `protocols` file is:

```
dn: cn=tcp, ou=protocols, ou=Services, dc=France, dc=airius, dc=com
cn: tcp
cn: tcP_Protocol
ipProtocolNumber: 0
description: Transmission Control Protocol
objectClass: top
objectClass: ipProtocol
```

rpc Mapping

An entry in the `rpc` file is usually of the form:

```
programname programnumber protocolalias #description
```

For example, the `rpc` file in the domain `France.airius.com` contains the following line:

```
yppasswdd 100123 yppasswd
```

The directory entry created from this line in the `rpc` file is:

```
dn: cn=yppasswdd, ou=rpc, ou=Services, dc=France, dc=airius, dc=com
cn: yppasswdd
cn: yppasswd
oncRpcNumber: 100123
objectClass: top
objectClass: oncRpc
```

ypservers Mapping

In the NIS environment, the `ypservers` file contains a list of the NIS servers in the domain.

For example, the `ypservers` file in the domain `France.airius.com` contains the following list:

```
brie
camembert
emmental
gorgonzola
roquefort
```

A directory entry is created for each server listed in the file. For example, the entry created for the server `brie` is:

```
dn: cn=brie, ou=ypservers, ou=Services, dc=France, dc=airius, dc=com
cn: brie
objectClass: top
objectClass: sunNisServer
```

NIS Schema

The schema is the set of rules that describe the data that can be stored in the directory. It defines the type of entries, their structure and their syntax. This section describes the NIS object classes and attributes supplied with the Solaris Extensions for Netscape Directory Server 4.11.

The object classes and attributes in the NIS schema are compliant with RFC 2307 *An Approach for Using LDAP as a Network Information Service*.

The standard NIS schema objects are included in the `slapd.oc.conf` and `slapd.at.conf` files used by the directory server. By default, these files are stored in `NSHOME/slapd-serverID`, where `NSHOME` represents the directory where you installed the server, and `serverID` represents the identifier you assigned to the server during the installation process.

Some specific object classes and attributes that are not part of the RFC are also needed to operate the NIS service. These objects are defined in the `nis.oc.conf` and `nis.at.conf` files stored by default in the directory `/opt/SUNWconn/ldap/default/schema`.

For more details on the directory schema, refer to the *Netscape Directory Server Administrator's Guide*.

NIS Object Classes

automount

Description: Defines an entry representing an NIS automount record.

Superior object class: top

Mandatory attributes: cn (commonName), automountInformation

Optional attribute: description

bootableDevice (auxiliary object class)

Description: Defines an entry representing any device that requires boot parameters. Used to import information from the `/etc/bootparams` file. It is an *auxiliary* object class, which means that it should be used in conjunction with a structural object class.

Superior object class: device

Mandatory attribute: cn (commonName)

Optional attributes: bootFile, bootParameter

ieee802Device (auxiliary object class)

Description: Defines entries representing any device that has a MAC address. Used to import information from the `/etc/ethers` file. It is an *auxiliary* object class, which means that it should be used in conjunction with a structural object class.

Superior object class: device

Optional attribute: `macAddress`

ipHost (auxiliary object class)

Description: Used to describe a device that has an IP address. Used to import information from the `/etc/hosts` file. It is an *auxiliary* object class, which means that it should be used in conjunction with a structural object class.

Superior object class: top

Mandatory attributes: `cn` (`commonName`), `ipHostNumber`

Optional attributes: `bootFile`, `bootParameter`, `description`, `l` (`localityName`), `macAddress`, `manager`, `serialNumber`

ipNetwork

Description: Defines an entry that describes an IP network. Used to import information from the `/etc/networks` file.

Superior object class: top

Mandatory attributes: `cn` (`commonName`), `ipNetworkNumber`

Optional attributes: `description`, `ipNetmaskNumber`, `l` (`localityName`), `manager`

ipProtocol

Description: Defines an entry that describes an IP protocol. Used to import information from the `/etc/protocols` file.

Superior object class: top

Mandatory attributes: `cn` (`commonName`), `ipProtocolNumber`

nisMailAlias

Description: Defines an entry that represents an NIS `mail.aliases` record. Used to import information from the `/etc/mail/aliases` file.

Superior object class: `top`

Mandatory attribute: `cn` (`commonName`)

Optional attribute: `rfc822MailMember`

nisMap

Description: Defines an entry that represents an NIS map.

Superior object class: `top`

Mandatory attribute: `nisMapName`

Optional attribute: `description`

nisNetGroup

Description: Defines an entry that represents an NIS `netgroup` record. Used to import information from the `/etc/netgroup` file.

Superior object class: `top`

Mandatory attribute: `cn` (`commonName`)

Optional attributes: `description`, `memberNisNetgroup`, `nisNetgroupTriple`

nisNetId

Description: Defines an entry that represents an NIS `netid.byname` record.

Superior object class: `top`

Mandatory attribute: `cn` (`commonName`)

Optional attribute: `nisNetIdGroup`, `nisNetIdHost`, `nisNetIdUser`

nisObject

Description: Defines an entry in the directory that represents an entry in an NIS map. The NIS key is stored in the cn attribute.

Superior object class: top

Mandatory attribute: nisMapName

Optional attributes: cn (commonName), description, nisMapEntry

nisSunObject

Description: Defines an entry in the directory that represents an entry in an NIS map. This object class is used in the generic NIS map definition in Sun Directory Services. The NIS key is stored in the sunNisKey attribute.

Superior object class: top

Mandatory attribute: nisMapName

Optional attributes: cn (commonName), description, nisMapEntry, sunNisKey

oncRpc

Description: Defines an entry that represents an Open Network Computing (ONC) remote procedure call (RPC). Used to import information from the `/etc/rpc` file.

Superior object class: top

Mandatory attributes: cn (commonName), oncRpcNumber

Optional attribute: description

posixAccount (auxiliary object class)

Description: Represents an account defined by POSIX attributes. Used to import information from the `/etc/passwd` file. It is an *auxiliary* object class, which means that it should be used in conjunction with a structural object class.

Superior object class: top

Mandatory attributes: cn (commonName), uid (userid), uidNumber, gidNumber, homeDirectory

Optional attributes: description, gecos, loginShell, userPassword

posixGroup

Description: Defines an entry that represents a group of POSIX accounts. Used to import information from the `/etc/group` file.

Superior object class: top

Mandatory attributes: cn (commonName), gidNumber

Optional attributes: description, memberUid, userPassword

shadowAccount (auxiliary object class)

Description: Used to represent a user that has a shadow password. It is an *auxiliary* object class, which means that it may be used in conjunction with any object class.

Superior object class: top

Mandatory attribute: uid (userid)

Optional attributes: description, shadowLastChange, shadowMax, shadowMin, shadowWarning, shadowInactive, shadowExpire, shadowFlag, userPassword

sunNisMap

Description: Used internally by the server to manage NIS maps. An entry is created under the `ou=admin,ou=Services` subtree for each map stored in the LDAP directory.

Superior object class: top

Mandatory attributes: sunNisDomain, sunNisMapFullName, sunNisMapState, sunNisMaster, sunNisSecurityMode

Optional attributes: description, seeAlso, sunNisDbmCache, sunNisDnsForwarding, sunNisInputFile, sunNisOutputName, sunNisLoadMap

sunNisServer

Description: Defines an entry that represents an NIS ypservers record. Used to import information from the `ypservers` file.

Superior object class: `top`

Mandatory attributes: `cn` (`commonName`)

Optional attribute: `description`

NIS Attributes

All attributes defined in the NIS schema have one of the following syntaxes:

- Distinguished name (`dn`)
- Case-ignore string (`cis`); An alphanumeric string, not case-sensitive
- Case-exact string (`ces`); A case-sensitive alphanumeric string
- Telephone number (`tel`)
- Integer (`int` or `long`)
- Binary (`bin`)
- Encrypted (`protected`); A value that has been encrypted using the method specified through the Admin Console. Possible values are **sunds**, **crypt**, or **none**.
- UTC time (`utctime`)

The following list of attributes in the NIS schema gives the attribute syntax, any alternative names, and explains how the attribute is used.

automountInformation

Description: The automount information for the entry in the NIS automount map.

Syntax: `ces`

bootFile

Description: The name of the file containing the boot parameters for the bootable device described by the entry.

Syntax: ces

bootParameter

Description: A boot parameter for the bootable device described by the entry.

Syntax: ces

commonName

Alias: cn

Description: The full name of the user described by the entry.

Syntax: cis

description

Description: The description of the entry object.

Syntax: cis

gecos

Description: The gecos field of the user described in the entry. Usually the user's common name. This attribute is single-valued.

Syntax: cis

gidNumber

Description: An integer that uniquely identifies a group in an administrative domain. This attribute is single-valued.

Syntax: long

homeDirectory

Description: Specifies the home directory of the user described by the entry. This attribute is single-valued.

Syntax: ces

ipHostNumber

Description: Specifies the IP address of the host described by the entry, in dotted decimal format, without leading zeros.

Syntax: cis

ipNetmaskNumber

Description: Specifies an IP netmask, in dotted decimal format, without leading zeros. This attribute is single-valued.

Syntax: cis

ipNetworkNumber

Description: Specifies the number of the IP network described by the entry, in dotted decimal format, without leading zeros. This attribute is single-valued.

Syntax: cis

ipProtocolNumber

Description: Specifies the port number/port type (for example UDP or TCP) pair for the IP protocol described by the entry. This attribute is single-valued.

Syntax: long

localityName

Alias: l

Description: The geographical locality of the object described by the entry.

Syntax: cis

loginShell

Description: The path to the login shell of the user described by the entry. This attribute is single-valued.

Syntax: ces

macAddress

Description: Specifies a MAC address for the device described by the entry, expressed in colon-separated hex notation.

Syntax: cis

manager

Description: The distinguished name of the manager of the person or object described by the entry.

Syntax: dn

memberNisNetgroup

Description: The name of a member of the netgroup described by the entry.

Syntax: cis

nisMapEntry

Description: Contains a record in the NIS map described by the entry. This attribute is single-valued.

Syntax: ces

nisMapName

Description: Specifies the name of the NIS map described by the entry. This attribute is single-valued.

Syntax: ces

nisNetgroupTriple

Description: Represents a triple of the form hostname/username/domainname associated with the netgroup described by the entry.

Syntax: cis

nisNetIdGroup

Description: Represents the group id associated with a record in the netid.byname map.

Syntax: ces

nisNetIdHost

Description: Represents the hostname associated with a record in the netid.byname map.

Syntax: ces

nisNetIdUser

Description: Represents the user id associated with a record in the netid.byname map.

Syntax: ces

oncRpcNumber

Description: RPC number of the RPC service described by the entry. This attribute is single-valued.

Syntax: long

rfc822MailMember

Description: Stores the email addresses (RFC-822 format) defined for members of the list.

Syntax: ces

seeAlso

Description: The distinguished name of an entry that contains information that is also of interest to anyone interested in the object described by this entry.

Syntax: dn

serialNumber

Description: The serial number of the device described by the entry.

Syntax: cis

shadowLastChange

Description: Indicates the number of days between January 1, 1970 and the day when the user password was last changed in the `/etc/shadow` file. This attribute is single-valued.

Syntax: long

shadowExpire

Description: Indicates the date on which the user login will be disabled. This attribute is single-valued.

Syntax: long

shadowFlag

Description: Reserved attribute, not currently in use.

Syntax: long

shadowInactive

Description: Indicates the number of days of inactivity allowed for the user. This attribute is single-valued.

Syntax: long

shadowMax

Description: Indicates the maximum number of days for which the user password remains valid. This attribute is single-valued.

Syntax: long

shadowMin

Description: Indicates the minimum number of days required between password changes. This attribute is single-valued.

Syntax: long

shadowWarning

Description: The number of days of advance warning given to the user before the user password expires. This attribute is single-valued.

Syntax: long

sunNisDbmCache

Description: Indicates whether the NIS/LDAP server must maintain the map described by the entry in plain NIS format. The possible values of this attribute are Enabled or Disabled. It is usually Enabled for a master server, and Disabled for a slave server. The value automatically changes to Disabled for a map that exceeds 50 000 entries. This attribute is single-valued.

Syntax: cis

sunNisDnsForwarding

Description: Indicates that the server must look up DNS for hostnames and addresses not found in the NIS tables. The possible values of this attribute are Enabled or Disabled. This attribute is created with the value Enabled when you run the `dsypinit` command with option `-b`. This attribute is single-valued.

Syntax: cis

sunNisDomain

Description: Gives the name of the NIS domain to which the map described by the entry belongs. This attribute is single-valued.

Syntax: ces

sunNisInputFile

Description: Stores the value of a special NIS key called `YP_INPUT_FILE`. This attribute is single-valued.

Syntax: ces

sunNisKey

Description: Stores the value of a NIS key (case-sensitive).

Syntax: ces

sunNisLoadMap

Description: Adding this attribute to the entry launches a reload of the map described by the entry. It builds the map from the entries already present in the directory. This attribute is single-valued, and you can give it any value. This attribute is automatically removed when the reload of the map is complete. Creating this attribute is equivalent to running the `dsypinit` command with option `-l`.

Syntax: cis

sunNisMapFullName

Description: Gives the full name of a Sun NIS map with the domain name as suffix. This attribute is single-valued.

Syntax: ces

sunNisMapState

Description: Indicates whether the map described by the entry is supported by the server. The possible values of this attribute are Enabled and Disabled. Enabled indicates that the map is supported by the server, Disabled that it is not. This attribute is single-valued.

Syntax: cis

sunNisMaster

Description: Specifies the hostname of the master server for the map described by the entry. This attribute is single-valued.

Syntax: ces

sunNisOutputName

Description: Stores the value of a special NIS key called YP_OUTPUT_FILE. This attribute is single-valued.

Syntax: ces

sunNisSecurityMode

Description: Sets the security mode for the map described by the entry. The possible values of this attribute are Secure and Insecure. When set to Secure, the server will accept connections from secure networks only. This attribute is single-valued. Setting this attribute to Secure is equivalent to running the `dsypinit` command with the option `-r`.

Syntax: cis

uidNumber

Description: An integer that uniquely identifies a user in an administrative domain. This attribute is single-valued.

Syntax: long

userid

Alias: uid

Description: The user ID of the user described by the entry.

Syntax: cis

userPassword

Description: The password that the user described by the entry uses to gain access to the entry. The directory server automatically crypts this attribute.

NIS/LDAP Mapping Summary

Table 4.1 gives a summary of the type and location of LDAP entries for each NIS map. For details on the mapping definition for each NIS file, refer to the appropriate section under “NIS File Entries/LDAP Entries” on page 72.

Table 4.1 NIS/LDAP Mapping Summary

NIS Source File	NIS Maps	Object Class of LDAP Entries	LDAP Subtree
/etc/mail/aliases	mail.aliases, mail.byaddres	nisMailAlias	ou=Aliases, ou=services
/etc/bootparams	bootparams	bootableDevice	ou=Hosts, ou=Services
/etc/ethers	ethers.byaddr, ethers.byname	ieee802Device	ou=Hosts, ou=Services
/etc/group	group.byname, group.bygid	posixGroup	ou=Group, ou=Services
/etc/hosts	hosts.byaddr, hosts.byname	ipHost	ou=Hosts, ou=Services
/etc/netgroup	netgroup	nisNetGroup	ou=Netgroup, ou=Services
	netgroup.byhost	nisObject	ou=netgroup.byhost, ou=Services
	netgroup.byuser	nisObject	ou=netgroup.byuser, ou=Services
/etc/networks	networks.byname, networks.byaddr	ipNetwork	ou=Networks, ou=Services
/etc/passwd	passwd.byname, passwd.byuid	posixAccount	ou=People

NIS/LDAP Mapping Summary

NIS Source File	NIS Maps	Object Class of LDAP Entries	LDAP Subtree
<code>/etc/protocols</code>	<code>protocols.byname,</code> <code>protocols.bynumber</code>	<code>ipProtocol</code>	<code>ou=Protocols, ou=Services</code>
<code>/etc/rpc</code>	<code>rpc.bynumber</code>	<code>oncRpc</code>	<code>ou=rpc, ou=Services</code>
<code>/var/yp/binding</code> <code>/domainName</code>	<code>ypservers</code>	<code>sunNisServer</code>	<code>ou=ypservers, ou=Services</code>
All other maps in <code>Makefile</code>	<code>mapName</code>	<code>nisObject</code>	<code>ou=mapName, ou=Services</code>

NIS Command & File Reference

This chapter provides reference information on the NIS daemons, commands and files. It covers standard NIS components as well as the NIS features delivered with the Solaris Extensions for Netscape Directory Server 4.11.

Deja.properties

Synopsis

The location of the `Deja.properties` file is:

```
/opt/SUNWconn/ldap/html/Deja.properties
```

Description

The `Deja.properties` file determines the display characteristics of Deja. It also defines the templates that are used to create and modify certain directory entries, such as NIS and RADIUS entries.

You must be authenticated as `superuser` or `root` to modify the `Deja.properties` file. When you have made modifications to this file, you must restart `Deja` for the modifications to take effect.

File Structure

The `Deja.properties` file consists of four sections:

- “General Properties” on page 101
- “Standard LDAP Properties” on page 102
- “NIS Properties” on page 104
- “RADIUS Properties” on page 106

File Syntax

Each section in the `Deja.properties` file contains a list of definitions. Each definition ends with a carriage return. The different elements in a definition are separated by commas. Related elements are separated by semi-colons.

For example, the attributes returned in RADIUS searches are defined as follows:

```
RADIUS_RU_LIST.default=      cn;RADIUS_RU_CN_ATTR_LABEL,  
uid;RADIUS_RU_UID_ATTR_LABEL
```

In this example, the definition is composed of two elements, separated by a comma. Each element consists of an attribute type (`cn` and `uid` in this example), and a label that is displayed in `Deja`, in the results table header row.

This example does not show the actual labels that appear in `Deja`’s menus. These are defined separately, in the *localized resource bundle*. The localized resource bundle contains translations in every supported locale for the user interface of `Deja`.

Labels

Standard `Deja` labels and identifiers (parameters ending in `_LABEL`, `_IDENTIFIER` or `_CHOICE`) are defined in the localized resource bundle. You cannot change these definitions. You can, however, create your own labels.

For example, if you want to the `ipHostNumber` attribute type to the list returned by default in a search on RADIUS remote users, you might modify the `RADIUS_RU_LIST.default` definition as follows:

```
RADIUS_RU_LIST.default=      cn;RADIUS_RU_CN_ATTR_LABEL,
uid;RADIUS_RU_UID_ATTR_LABEL, ipHostNumber;Host Number
```

This definition is local to your `Deja.properties` file. It is not part of the localized resource bundle.

User Input

In the `Deja.properties` file, user input is represented using the character sequence `{0}`. For example, in a search filter, the definition `(cn={0}*)` specifies that the search will result in entries for which `cn` contains the search string.

The character sequence `{definition$}` is used by `Deja` to define a user input field in searches. The expression `definition`, can consist of the following elements:

- Attribute name - the type of attribute you are searching for
- Label - a text string to appear by the input field
- Field type - the type of input field required. The field type can have one of the following values: **crypt**, **string**, **int** or **ipaddr**. **crypt** is a text field where each typed character is replaced with *. **string** is a text field that accepts any character. **int** is a text field that accepts only integer numbers. **ipaddr** consists of four *int* fields.

If Field type is not specified, the string input field is used by default. For example, the following expression `{iphonenumber;IP Host Number;ipaddr$}` generates an `ipaddr` input field with the label `IP Host Number`. It also specifies that the user input is an attribute of the type `iphonenumber`.

General Properties

SCHEMA_THREAD_TIME_LIMIT

Defines a time limit in milliseconds on the time it takes `Deja` to read the schema. The default value is no time limit.

BROWSER_ENTRY_LIMIT

Specifies the maximum number of entries that can be displayed in the browser. If a limit has been set, you must refresh certain subtrees before opening more. The default value is no limit.

BROWSER_SUBENTRY_LIMIT

Defines the maximum number of immediate children of an entry that can be displayed in the browser. The default value is no limit.

BROWSER_LOAD_SUBNODES_TIME_LIMIT

Specifies the maximum amount of time allowed for Deja to load the children of a node when the node is opened in the browser. This is not the amount of time it then takes to display those children. The default value is 10000 milliseconds.

BROWSER_CHECK_NODE_TIME_LIMIT

This is the maximum time taken for Deja to verify whether an entry is a leaf or a node. The default value is 2000 milliseconds.

STANDARD_SECURITY_AUTHENTICATION

Defines the standard authentication mechanism used in the login panel. The only possible value for this parameter is **simple**.

Standard LDAP Properties

In this section the following tokens are defined:

STANDARD_ATTRIBUTES_CRYPTED

In the View, Modify and Create windows of Deja, some attribute values are not displayed, or replaced by a localized text string. You can specify the attributes you want to be hidden by adding them to the `STANDARD_ATTRIBUTES_CRYPTED` list. Attribute names are separated by commas. By default the values for `userpassword`, `radiusppppasswd`, `radiusloginpasswd`, `chappassword`, and `radiusslippasswd` are hidden.

STANDARD_LOGIN_SEARCH_FILTER

The search feature of the login panel operates using the filter defined with this label. By default it is `(| (cn=*{0}*) (uid=*{0}*))`. This search filter means that either the `cn` attribute or the `uid` attribute should contain the search string typed by the user in the search text field.

STANDARD_LOGIN_MAX_SEARCH_RESULT

Specifies the maximum number of search results per naming context returned by a login search. The default value is 55.

STANDARD_LOGIN_ALIASES

Defines an alias for the user DN you use to login to Deja. By default, there are no aliases defined, and the STANDARD_LOGIN_ALIASES parameter is commented out. The definition in the `Deja.properties` file reads as follows:

```
# STANDARD_LOGIN_ALIASES= userA_alias; userA_dn; userB_alias; userB_dn
```

To add a login alias, you must uncomment the line, add an alias name and a user DN for login. For example, if the user `cn=Robert Travis,ou=sales,o=sun,c=us` wants to login frequently, you can create an alias for him, for example, `rob`. To add this alias, you would edit the STANDARD_LOGIN_ALIASES definition in the `Deja.properties` file to read as follows:

```
STANDARD_LOGIN_ALIASES= rob; cn=Robert Travis,ou=sales,o=sun,c=us
```

Note If you create several aliases, you must use a semi-colon to separate them, and not a comma, which is the standard syntax, because the comma is used to separate the different elements in the DN. The semi-colon separates the elements of a DN from a new alias definition.

For example, if you also wanted to add an alias for the NIS administrator whose DN is `cn=NIS Manager,o=sun,c=us`, the STANDARD_LOGIN_ALIASES definition in the `Deja.properties` file would read as follows:

```
STANDARD_LOGIN_ALIASES= rob ; cn=Robert Travis,ou=sales,o=sun,c=us ; NIS
admin; cn=NIS Manager, o=sun, c=us
```

When Deja is restarted the aliases are available in the Login panel. This parameter is case-sensitive.

STANDARD_SEARCH_FILTERS

Specifies the standard searches available in Deja. Each entry in this list is defined on a separate line.

STANDARD_SEARCH_FILTER_name

Defines each search available, where *name* is the name of the search specified in STANDARD_SEARCH_FILTERS. A search definition consists of the search name (for example, STANDARD_SEARCH_FILTER_PERSON), the label that appears in the Search Type option button (for example, STANDARD_SEARCH_FILTER_PERSON_IDENTIFIER), and the search definition (for example, `(&(objectclass=*)(cn={0}*))`).

STANDARD_SEARCH_TABLE_LABELS

Contains a list of the attributes and header labels for the search results table. By default the cn, telephoneNumber and mail attributes are listed.

STANDARD_CREATE_PASTE_CLEAR_DATA

When you paste an entry to the Create panel, the paste works in one of two ways:

- The paste action can remove information from the Create panel before pasting the entry.
- The paste action does not clear data from the Create panel before pasting. This is useful when you want to create an entry that contains the characteristics of two or more entries.

This property specifies the type of paste. It can be set to **true** or **false**. **true** indicates that data is cleared from the entry before pasting. By default this parameter is set to **false**. The lines in the `Deja.properties` file that define this property are as follows by default:

```
# Standard Create
STANDARD_CREATE_PASTE_CLEAR_DATA=FALSE
#STANDARD_CREATE_PASTE_CLEAR_DATA=TRUE
```

To change the setting, uncomment the line you want, and comment out the other.

NIS Properties

NIS_MAPS

Specifies the list of maps available in Deja. Each map name is followed by a semicolon and the label that appears in the Map Name option button of the NIS Search, Create or Modify panels. If you create a new map in the `nis.mapping` file, you must declare the map name in the `NIS_MAPS` token in the `Deja.properties` file. The syntax is:

```
NIS_MAPS= map.name;map_label, map.name;map_label, ...
```

NIS_FILTER.map.name

Specifies the filter that is used in the NIS Search panel. This definition is automatically generated by running `dejasync`.

NIS_DOMAIN.map.name

Specifies the label that appears in the NIS Create, Modify and Search panels. It shows to which domain the NIS map applies. This definition is automatically generated by running `dejasync`.

NIS_NAMINGATTR.map.name

Specifies the naming attributes that are available in the NIS Create panel. This is a comma separated list. This definition is automatically generated by running `dejasync`.

NIS_ROOT.map.name

Specifies the DN of the root entry used for NIS searches. It is also the default parent entry displayed in the NIS Create panel. This definition is automatically generated by running `dejasync`.

NIS_OCLASS.map.name

Specifies the default object classes that are added to an entry definition in the NIS Create Panel. This is a comma separated list. This definition is automatically generated by running `dejasync`.

NIS_LIST.map.name

Contains names of the attributes and header labels for the NIS search results table. The syntax is:

```
NIS_LIST.map.name= attribute;header_label, attribute;header_label, ...
```

NIS_ADD.map.name

Specifies labels and syntax for attributes in the NIS Create panel. The syntax is:

```
NIS_ADD.map.name= attribute;label:syntax, attribute;label:syntax, ...
```

Where syntax is one of the four basic input types (**int**, **string**, **crypt** and **ipaddr**). If a syntax isn't specified, the default value, **string**, is used. Specifying a syntax is useful to constrain user input:

- **crypt** is a text field where each character typed in is replaced with *.
- **string** is a text field that accepts any character.
- **int** is a text field that accepts only integer numbers.
- **ipaddr** consists of four **int** fields, in the format *int.int.int.int*.

NIS_LIST.default

Contains the names of the attributes listed in the NIS search results table if **NIS_LIST** is not defined for a map.

RADIUS Properties

`RADIUS_RU_SEARCH, RADIUS_RAS_SEARCH`

Specifies the standard searches available in Deja for remote users (RU) and remote access servers (RAS). Each entry in this list is defined on a separate line. The syntax is:

`RADIUS_RU_SEARCH= Name;label, Name;label, ...`

Where *Name* is the name of the search, and *label* is the text that appears in the Search Type option button.

`RADIUS_RU_FILTER.Name, RADIUS_RAS_FILTER.Name`

Defines the search filter used in the search, where *Name* is the name of the search specified in `RADIUS_RU_SEARCH` or `RADIUS_RAS_SEARCH`.

`RADIUS_RU_LIST.Name, RADIUS_RAS_LIST.Name`

Contains a list of the attributes and header labels for the search results table.

`RADIUS_RU_LIST.default, RADIUS_RAS_LIST.default`

Contains the default list of the attributes and header labels for the search results table if a `RADIUS_RU_LIST.Name` or `RADIUS_RAS_LIST.Name` definition does not exist for the search.

`RADIUS_COMPLEX_SEARCH_LIST`

Contains a list of the attributes and header labels for the complex searches results table.

`RADIUS_RU_ADD_COMMON, RADIUS_RAS_ADD_COMMON`

Specifies alternative names for attributes that are displayed in the Choose Attributes list of the RADIUS Create panel. The syntax is:

`RADIUS_RU_ADD_COMMON= attribute;label:type`

Where *attribute* is the name of an attribute, *label* is the name you want to appear in the Choose Attributes list, and *type* is the input type. You can restrict user input to one of the four basic input types (**int**, **string**, **crypt** or **ipaddr**). The default type is **string**.

RADIUS_RU_PROFILE, RADIUS_RAS_PROFILE

Three RADIUS Remote User profiles are defined in the default `Deja.properties` file. You can add more profiles, or add attributes to the existing profiles, but you should not remove default attributes in the existing profiles.

`RADIUS_RU_PROFILE` and `RADIUS_RAS_PROFILE` specify the RADIUS profiles available to Deja. The default profiles are SLIP, PPP and LOGIN. The syntax is:

```
RADIUS_RU_PROFILE= profile_name;label, profile_name;label ...
```

Where *profile_name* is the name of the profile, and *label* is the label that appears in the Create or Modify panels.

RADIUS_RU_ADD.Name, RADIUS_RAS_ADD.Name

Defines the default attributes that are added to the entry automatically. The syntax is:

```
RADIUS_RU_ADD.profile_name= attribute;label;input_type, ...
```

Where *attribute* is the attribute you want automatically added to the entry definition, *label* is the name to appear in the entry definition, and *input_type* is one of the four basic input types (int, string, crypt or ipaddr). The default *input_type* is string.

RADIUS_RU_OCLASS

Specifies the object class associated with the RADIUS remote user entry type. A single object class is required for each type. This definition is automatically updated if you use the `dejasync` utility. The default object class is `remoteUser`.

RADIUS_RAS_OCLASS

Specifies the object class associated with the RADIUS remote access server entry type. A single object class is required for each type. This definition is automatically updated if you use the `dejasync` utility. The default object class is `nas`.

RADIUS_RU_ROOT

Specifies the DN of the root entry used for RADIUS remote user searches. It is also the default parent entry displayed in the RADIUS Create panel. This definition is automatically updated if you use the `dejasync` utility. The default value is `o=xyz_remote_users,c=us`.

dejasync

RADIUS_RAS_ROOT

Specifies the DN of the root entry used for RADIUS remote access server searches. It is also the default parent entry displayed in the RADIUS Create panel. This definition is automatically updated if you use the `dejasync` utility. The default value is `o=xyz_ras,c=us`.

RADIUS_RU_NAMINGATTR

Specifies the naming attributes that are available in the RADIUS Create panel for remote user entries. This is a comma separated list. The default naming attributes are `cn` and `uid`.

RADIUS_RAS_NAMINGATTR

Specifies the naming attributes that are available in the RADIUS Create panel for remote access server entries. This is a comma separated list. The default naming attribute is `cn`.

RADIUS_MAX_FAIL

Specifies the search limit for the RADIUS remote user blocked accounts search. The blocked accounts search returns entries that have a value for the attribute `radiusAuthFailedAccess` that is greater than or equal to the value of `RADIUS_MAX_FAIL`. The default value is 1. This definition is automatically updated if you use the `dejasync` utility.

See Also

See “`dsysync`” on page 128, “`nis.mapping`” on page 134.

dejasync

Synopsis

The command syntax for `dejasync` is:

```
/opt/SUNWconn/ldap/sbin/dejasync [-v] [-d Deja_properties_directory]  
[-n NIS_mapping_file] [-r RADIUS_mapping_file]
```

Description

dejasync is a command line utility that synchronizes the `Deja.properties` files with the NIS and RADIUS mapping files (`nis.mapping` and `radius.mapping`) on the directory server. Use it when you have made modifications to the mapping files and you want the changes to be carried over into Deja. As necessary, it creates or updates tokens in the `Deja.properties` file.

dejasync also backs up the `Deja.properties` file.

You must be logged in as `root` or `superuser` to run `dejasync`.

`nis.mapping` File

dejasync gets from the `Deja.properties` file the list of maps managed by Deja for NIS. These are lines that start with the `NIS_MAPS` token.

For each map in the `Deja.properties` file, `dejasync` creates a new map definition by copying the following tokens from the `nis.mapping` file into the `Deja.properties` file:

- `NIS_FILTER`
- `NIS_DOMAIN`
- `NIS_ROOT`
- `NIS_NAMINGATTR`
- `NIS_OCLASS`

If these tokens exist in the `Deja.properties` file, the `dejasync` command updates them. If they do not exist, it creates them.

`radius.mapping` File

When synchronizing `Deja.properties` with the `radius.mapping` file, `dejasync` copies the `Max_allowed_failures`, `base-DN` and `FILTER` tokens from the `radius.mapping` file to the `Deja.properties` file:

- `RADIUS_RU_OCLASS`
- `RADIUS_RAS_OCLASS`
- `RADIUS_RU_ROOT`
- `RADIUS_RAS_ROOT`
- `RADIUS_MAX_FAIL`

dsexport

If these tokens exist in the `Deja.properties` file it updates them. If they do not exist it creates them.

Options

`-v`

Enables verbose mode.

`-d Deja_properties_directory`

Specifies the directory containing the `Deja.properties` file. By default this is `/opt/SUNWconn/ldap/html`.

`-n NIS_mapping_file`

Specifies the filename of the NIS mapping file. By default this is `/etc/opt/SUNWconn/ldap/current/mapping/nis.mapping`.

`-r RADIUS_mapping_file`

Specifies the filename of the RADIUS mapping file. By default this is `/etc/opt/SUNWconn/ldap/current/mapping/radius.mapping`.

See Also

See “`nis.mapping`” on page 134, and “`Deja.properties`” on page 99.

dsexport

Synopsis

The command syntax for `dsexport` is:

```
/opt/SUNWconn/ldap/sbin/dsexport [-c dsserv_conf_file] [-d debuglevel]  
[-D binddn] [-f frontend] [-h host] [-m mappingfile] [-M bindmethod] [-p  
port] [-S schema_entry_dn] -t table [-T ldap_timeout] [-V  
variable=value]... [-w password] [outputfile]
```

Description

The `dsexport` command exports directory entries using the specified mapping file to restore the formatted files which were used to create the entries in the first place.

If you remove the Solaris Extensions for Netscape Directory Server 4.11 and want to restore a standard NIS service, you can run `dsexport` to restore and bring up to date the NIS source files used by the NIS service to build NIS maps. Some comments and formatting information that were in the original file are lost. This information is not stored in the directory server. The lines may also not be in the original order.

The exported file is displayed on standard output unless you specify an output file name.

Options

`-c dsserv_conf_file`

Specifies the location of the directory server configuration file. The default is `NSHOME/slapd-serverID/config/slapd.conf`.

`-d debuglevel`

This option tells the directory server daemon at what level information should be logged to log files. You can request any combination of the following levels:

Mask	Description
1	Trace
2	Packets
4	Arguments
8	Connections
16	BER
32	Filters
64	Configuration
128	Access control
256	Statistics (summary level)

dsexport

Mask	Description
512	Statistics (detailed level)
1024	Not used
2048	Parse
65535	All information

To request more than one category of debugging information, add the masks. For example, to request trace and filter information, specify a debuglevel of 33.

-D *binddn*

Specifies the DN used for authenticating the administrator who is connecting to the directory. By default, this information is taken from the configuration file.

-F *frontend*

Specifies the relevant Front-End section in the mapping file. The default value of *front_end* is NIS.

-h *host*

Indicates the hostname of the directory server that holds the data store from which you want to export the data. The default is the local host.

-m *mappingfile*

Specifies the mapping file that the `dsexport` command uses to export entries from the LDAP directory. The default mapping file is `/etc/opt/SUNWconn/ldap/current/mapping/nis.mapping`.

-M *bindmethod*

Specifies the type of bind and the authentication mechanism used to bind to the directory. The only possible value for *bindmethod* is **simple**.

-P *port*

Specifies the UDP port on which the `slapd` directory server daemon resides. The default value is 389.

-t *table*

Specifies the exact name of the table in the Table section of the mapping file. Note that for the `nis.mapping` file only the tables which have a `LINE` definition in the `IMPORT` section can be used. For example, to export `hosts`, use the

`hosts.byaddr` Table section in the mapping file. For example, if you want to restore the `/etc/passwd` file, the table name in the default mapping file is `passwd.byname`.

`-S schema_entry_dn`

Specifies the DN used by `dsexport` to read the schema from the LDAP server. By default, the value of `schema_entry_dn` is `cn=schema`.

`-T ldap_timeout`

Specifies a timeout on the connection to the directory. The default value is 60 seconds.

`-V variable=value`

Used to set variables. You can repeat this option to specify multiple variable-value pairs. For example:

```
-V BASE_DN=o=sun,c=us -V foo=bar
```

`-w password`

Specifies the password used for authenticating the administrator who is connecting to the directory to perform changes. It is preferable not to use the `-w` option in a multi-user environment. This is because the password will be displayed in a listing of running processes. If you do not specify a bind DN and password, this information is read from the configuration file. If you specify a bind DN and not a password, you are prompted to provide one. These alternatives are more secure than supplying a password on the command line with the `-w` option.

`output_file`

Specifies the output file to use.

See Also

See “`nis.mapping`” on page 134, “`dsimport`” on page 114. For information on `slapd.conf`, refer to the *Netscape Directory Server Administrator's Guide*.

dsimport

dsimport

Synopsis

The command syntax for dsimport is:

```
/opt/SUNWconn/ldap/sbin/dsimport [-n] [-r] [-s] [-c dsserv_conf_file]  
[-d debuglevel] [-D binddn] [-f front_end] [-h host] [-m mapping_file]  
[-M bindmethod] [-p port] [-S schema_entry_dn] -t table [-T  
ldap-timeout] [-V variable=value]... [-w passwd] [file...]
```

Description

The dsimport utility creates directory entries from any text file in which one line corresponds to one directory entry. If the entry already exists, dsimport updates it.

You must create a mapping file that specifies the semantics of the information provided in each line of the input file. You might also need to create an LDAP object class and attributes that are specific to the type of information you want to store in the directory.

The dsimport utility is also used during the initialization of the NIS service to import all the information stored in NIS files into the LDAP directory. When you run the dsypinstall script to configure the directory server as an NIS server, the NIS information available on your server is automatically added to your directory database through a call to dsimport. The mapping of NIS files into LDAP object classes and attributes is provided in the `nis.mapping` file. For information on the NIS/LDAP information mapping, refer to Chapter 4, “NIS Information in the LDAP Directory.”

The mapping semantics and syntax accepted by the dsimport utility are described in Appendix A, “Mapping Syntax and Semantics.”

Options

-n

Do not create or modify directory entries but instead output them in LDIF format. You can then use the `ldapadd(1)` command to perform changes in the directory based on the LDIF files. The default is not to produce LDIF files and to create or modify directory entries directly.

-r

Remove entries in the directory that do not correspond to a line in one of the input files. The default is not to remove entries.

-s

Shift DN. Normally an entry is edited in place. With this option, however, if the DN of the imported entry is changed, its DN is modified. (The entry is "shifted" to a new location in the directory). For example, there is an entry in the LDAP directory with the DN `cn=joe,ou=sales,ou=people,o=airius,c=us`, and the `BASE_DN` for people is set to `ou=people,o=airius,c=us`. If Joe's entry did not exist, it would be created with the DN `cn=joe,ou=people,o=airius,c=us`. But since it already exists, the alternatives are:

1. Specify **-s** on the `dsimport` command line and change the DN so that Joe's entry is now where it would be if it was just created. Joe's entry is shifted from `cn=joe,ou=sales,ou=people,o=airius,c=us` to `cn=joe,ou=people,o=airius,c=us`.
2. Or do not modify the DN of the entry and just apply any necessary changes to the existing entry. Joe's entry stays at `cn=joe,ou=sales,ou=people,o=airius,c=us`.

-c *dsserv_conf_file*

Specifies the location of the directory server configuration file. The default is `NSHOME>/slapd-<serverID>/config/slapd.conf`.

-d *debuglevel*

This option tells the directory server daemon at what level information should be logged to log files. You can request any combination of the following levels:

Mask	Description
1	Trace
2	Packets
4	Arguments
8	Connections

dsimport

Mask	Description
16	BER
32	Filters
64	Configuration
128	Access control
256	Statistics (summary level)
512	Statistics (detailed level)
1024	Not used
2048	Parse
65535	All information

To request more than one category of debugging information, add the masks. For example, to request trace and filter information, specify a *debuglevel* of 33.

-D *binddn*

Specifies the DN used for authenticating the administrator who is connecting to the directory. By default, this information is taken from the configuration file.

-F *frontend*

Specifies the relevant Front-End section in the mapping file. The default value of *front_end* is NIS.

-h *host*

Indicates the hostname of the directory server that holds the data store from which you want to export the data. The default is the local host.

-m *mappingfile*

Specifies the mapping file that the `dsimport` command uses to export entries from the LDAP directory. The default mapping file is `/etc/opt/SUNWconn/ldap/current/mapping/nis.mapping`.

-M *bindmethod*

Specifies the type of bind and the authentication mechanism used to bind to the directory. The only possible value for *bindmethod* is **simple**.

-p *port*

Specifies the UDP port on which the `slapd` directory server daemon resides. The default value is 389.

`-t table`

Specifies the exact name of the table in the Table section of the mapping file.

`-S schema_entry_dn`

Specifies the DN used by `dsimport` to read the schema from the LDAP server. By default, the value of *schema_entry_dn* is `cn=schema`.

`-T ldap_timeout`

Specifies a timeout on the connection to the directory. The default value is 60 seconds.

`-V variable=value`

Used to set variables. You can repeat this option to specify multiple variable-value pairs. For example:

`-V BASE_DN=o=sun,c=us -V foo=bar`

`-w password`

Specifies the password used for authenticating the administrator who is connecting to the directory to perform changes. It is preferable not to use the `-w` option in a multi-user environment. This is because the password will be displayed in a listing of running processes. If you do not specify a bind DN and password, this information is read from the configuration file. If you specify a bind DN and not a password, you are prompted to provide one. These alternatives are more secure than supplying a password on the command line with the `-w` option.

file

Specifies one or several files that the `dsimport` command takes as input. If you do not specify a file on the command line, `dsimport` reads from standard input.

See Also

See “`nis.mapping`” on page 134, “`dsexport`” on page 110, “`dsypinstall`” on page 125.

For information on `slapd`, `slapd.conf`, and `ldapadd`, refer to the *Netscape Directory Server Administrator's Guide*.

dsyp

dsyp

Synopsis

The command syntax for `dsyp` is:

```
/etc/init.d/dsyp start | stop
```

Description

The `dsyp` command can be used to initialize or disable the NIS/LDAP synchronized service.

Options

start

Initializes NIS/LDAP synchronized service by stopping the `ypserv(1M)` and the `rpc.yppasswdd(1M)` daemons, and starting the `dsypserv` daemon.

stop

Disables NIS/LDAP synchronized service by stopping the `dsyppasswdd` and `dsypserv` daemons, and by calling the `ypstart(1M)` script.

See Also

See the man pages for `ypserv(1M)`, `rpc.yppasswd(1M)`, `ypstart(1M)`, and “`dsyppasswdd`” on page 126.

dsypaddmap

Synopsis

The command syntax for dsypaddmap is:

```
dsypaddmap [-b] [-l] [-r] [-d domain-name] [-f mapping-file] -m
nis_master [-n naming-context] map-name
```

Description

The dsypaddmap command simplifies the process of creating NIS-to-LDAP mapping definitions. When you want to add support for an NIS map in the LDAP directory, use dsypaddmap to:

- Modify the `nis.mapping` file to add a mapping definition for the map that you want to add to the LDAP directory
- Add an organizational unit (ou) subtree with the new map name under the `ou=services` subtree in the LDAP directory

Note The dsypaddmap command does not import data into the directory. To import the NIS data into the directory, use the `dsimport` command. See “dsimport” on page 114.

Options

-b

Inserts the `YP_INTERDOMAIN` key into the map. This key causes the NIS server to use DNS for hostname and address lookups for hosts not found in the maps.

-l

Initializes the specified map with LDAP entries already present in the directory. By default, only entries created or modified after the map declaration are put in the map. Security might be at risk if invalid entries are present in the directory database.

dsypdelmap

-r

Used to secure maps. With this option, the NIS server will only accept connections from the `root` user.

-d *domain-name*

Specifies the name of the domain to which the map belongs.

-f *mapping-file*

Specifies the path to the mapping file. By default it is `/etc/opt/SUNWconn/ldap/current/mapping`.

-m *nis-master*

Specifies the hostname of the master server for the map added to the LDAP directory.

-n *naming-context*

Specifies the subtree under which the LDAP entries corresponding to the new map will be created. By default, the entries are created under the subtree specified in the `NAMING_CONTEXT` variable in the `nis.mapping` file.

map-name

Specifies the name of the NIS map to be added to the LDAP directory.

See Also

See “dsypdelmap” on page 120.

dsypdelmap

Synopsis

The command syntax for dsypdelmap is:

```
dsypdelmap [-d domain-name] [-f mapping-file] [-n naming-context]  
map-name
```


Description

The `dsypdelmap` command can be used to individually disable NIS maps stored in the LDAP directory. This command does not remove the corresponding mapping definition from the `nis.mapping` file. It does not remove the corresponding entries from the directory either, because the directory entries might be shared with other applications. However, the map is not listed in the result of a `ypcat(1)`, `ypwhich(1)`, or `ypmatch()` command.

Options

`-d domain-name`

Specifies the name of the domain to which the map belongs.

`-f mapping-file`

Specifies the path to the mapping file. By default it is `/etc/opt/SUNWconn/ldap/current/mapping`.

`-n naming-context`

Specifies the subtree under which the LDAP entries corresponding to the map are located. By default, the entries are located under the subtree specified in the `NAMING_CONTEXT` variable in the `nis.mapping` file.

map-name

Specifies the name of the NIS map to be disabled in the LDAP directory.

See Also

See “`dsypaddmap`” on page 119.

dsypinit

dsypinit

Synopsis

The command syntax for `dsypinit` is:

```
/opt/SUNWconn/ldap/sbin/dsypinit -c  
/opt/SUNWconn/ldap/sbin/dsypinit -m [-r] [-b] [-l] [-k key_value] [-d  
domain] [table *]  
/opt/SUNWconn/ldap/sbin/dsypinit -s master_server  
/opt/SUNWconn/ldap/sbin/dsypinit -u [-d domain] [table *]
```

Description

The `dsypinit` command is used to set up the Netscape Directory Server as an NIS server. You can start the NIS server as a master server, a slave server, or even as an NIS client.

The `dsypinit` command also builds the NIS tables that are stored on the server.

If you need to set up both master and slave servers, you must run `dsypinit` on the masters before you run it on the slaves.

Options

The `dsypinit` command can be run in four modes:

- NIS client
- NIS master server
- NIS slave server
- Disable NIS tables

Client Option

-c

Used to set up an NIS client. You are prompted for a list of NIS servers. This list should be ordered from server which is physically closest to the client to the one which is the most far away. These servers must have an entry in the `/etc/hosts` file. The hosts file contains the names and IP addresses of all hosts on the network. The `dsypinit` command also keeps a list of hosts in the file `/var/yp/binding/domain/ypservers`. This file is used when the `ypbind` command is running without the `-broadcast` option.

Master Server Options

-m

Used to set up and build the tables for an NIS master server database.

-r

Used to secure maps. With this option, the NIS server will only accept connections from the `root` user.

-b

Specifies that the server will use the DNS resolver `dameon`, `dsyprsvd`, to look up DNS for hostnames and addresses not found in the NIS tables.

-l

Initializes the NIS tables with entries already present in the LDAP database. Security might be at risk if invalid entries are present in the database.

-k *key_value*

Used to insert a special key and value in the tables specified on the command line. Special keys are usually prefixed by `YP_`. They are interpreted by NIS utilities.

Keys include:

- `YP_MASTER_NAME` (set to `localhost`)
- `YP_INTERDOMAIN` (set with `-b` option)
- `YP_SECURE` (set with `-r` option)
- `YP_DOMAIN_NAME` (set with `-d` option)

dsypinit

- YP_OUTPUT_NAME
- YP_INPUT_FILE

Note It is NOT recommended to change these key directories by using `-k`.

`-d domain`

Restricts the tables initialized to those belonging to the specified domain. The other tables in the mapping file are not initialized. By default, all tables in all domains are initialized.

`table *`

Restricts the tables initialized to those specified. This option makes it possible to initialize some tables with different options if necessary.

Slave Server Options

`-s`

Used to set up a slave server database.

`master_server`

Specifies the master server from which the NIS tables are propagated. It must be an existing reachable NIS server.

Disable Options

`-u`

Used to disable the specified NIS tables in the specified domain. If the `-u` option is used without the `-d` option and without a list of tables, none of the NIS domains managed by the server are initialized.

`-d domain`

Specifies a domain for which the specified tables will be disabled. All other domains and tables managed by the server are normally initialized.

`table *`

Specifies the tables to be disabled. If no tables are specified, all the tables within the specified domain will be disabled.

See Also

See “dsypinstall” on page 125, “nis.mapping” on page 134, and the man pages for `ypbind(1M)` and `dsyprsvd(1M)`.

dsypinstall

Synopsis

The syntax for `dsypinstall` is:

```
/opt/SUNWconn/ldap/sbin/dsypinstall -u
```

Description

The `dsypinstall` script initializes the NIS server and imports NIS data into the LDAP database. This script is interactive, you are prompted for:

1. The name of the NIS domain managed by the server.
The name you provide is used to create the directory subtree under which all NIS entries are stored.
2. The installation directory for the Netscape Directory Server.
3. The DN of the Netscape Directory Server directory manager.
The DN you provide must be the same as the one you provided in the setup script for the Netscape Directory Server. This DN has all permissions on the Netscape Directory Server. By default, it is `cn=Directory Manager`.
4. The port number where the directory server listens for LDAP traffic.
5. The DN of the administrator for NIS information.
You must create an entry for the NIS administrator in the directory. You must also create an ACI giving the administrator all permissions on the NIS subtrees in the directory. Refer to “Access Control on NIS Information” on page 30.

dsyppasswdd

6. The location of the NIS source files.

The `dsypinstall` script assumes that your Makefile is located in `/var/yp`. It also assumes that the source files for NIS tables are all located in the directory that you specify when prompted, except for the `aliases` file which is assumed to be in `/etc/mail`.

7. A list of NIS servers.

The `dsypinstall` script calls `dsypinit` to initialize the NIS server, and it calls `dsimport` to import data from the NIS source files to the LDAP database. The NIS-to-LDAP information mapping is defined in the `nis.mapping` file.

After running the `dsypinstall` script, you must restart the Directory Server daemon, `ns-slapd`.

Options

`-u`

Used to remove all files and daemons that provide NIS synchronization with the Directory Server.

See Also

See “`dsimport`” on page 114, “`dsypinit`” on page 122, “`nis.mapping`” on page 134.

dsyppasswdd

Synopsis

The syntax for `dsyppasswd` is:

```
/opt/SUNWconn/ldap/lib/dsyppasswdd [-m] [-D pwdfilesdir] [-M directory]  
[-nopw] [-nogecos] [-noshell] [-nofiles]
```

Description

The dsyppasswdd daemon manages changes to the NIS passwd map stored in the LDAP directory. It is a modified version of the NIS `rpc.yppasswdd` daemon to run with Netscape Directory Server. It runs on the master server, and responds to NIS requests from remote users to change their password, shell, or `gecos` fields in the `passwd.byname` map.

The requested changes are actually made in the LDAP database and the NIS map is rebuilt from there.

Change requests are made using the `passwd` command.

Options

`-m`

Performs a push operation after each modification of the NIS passwd map.

`-D pwdfilesdir`

Indicates the name of the directory containing the `passwd` and `shadow` files used by NIS. The default is `/etc/yp`.

`-M directory`

Indicates the name of the directory containing the LDAP-to-NIS mapping file, `nis.mapping`. The default directory is `/etc/opt/SUNWconn/ldap/current/mapping`.

`-nopw`

Indicates that passwords may not be changed remotely using the `passwd` command.

`-nogecos`

Indicates that the field containing the user's real name and other mail-related information may not be changed remotely using the `passwd` command.

`-noshell`

Indicates that the user's shell may not be changed remotely using the `passwd` command.

dsypsync

`-nofiles`

Prevents the synchronization of `passwd` and `shadow` files; see the NOTES section.

Notes

Although the LDAP directory is presumed to be the master repository for password data, many legacy installations have scripts which change user passwords in the NIS source files and issue a `make` command. To ensure that these files are kept synchronized with the directory, the `dsyppasswdd` daemon copies any updates made by a client system into the standard NIS `passwd` and `shadow` files. These files are presumed to be in `/etc/yp` unless the `-D` option is used to specify otherwise.

Encrypted passwords are usually stored using the Unix `crypt` format.

A password change operation made via the `passwd(1)` command and the `dsyppasswdd` daemon will always replace any existing passwords in the directory with a single entry encrypted using `crypt`.

See Also

See the man pages for `passwd(1)`, `passwd(4)`, and `crypt(3C)`

dsypsync

Synopsis

The command syntax for `dsypsync` is:

```
/opt/SUNWconn/ldap/sbin/dsypsync [-d domain-name] [map-name]
```


Description

The `dsypsync` command is used to manually resynchronize NIS maps with the NIS information stored in the directory. You can resynchronize all the maps within a specified domain, or individual maps.

In normal operation, the synchronization is done automatically. You may need to resynchronize manually using `dsypsync` after a system crash or after rebuilding the LDAP database using the `ldif2db` command.

Options

`-d domain-name`

Specifies to resynchronize all the maps in the specified domain. If you do not specify either a domain name or a map name, by default, all maps in all domains controlled by the server are resynchronized.

`map-name`

Specifies the map which you want to resynchronize. If you do not specify either a domain name or a map name, by default, all maps in all domains controlled by the server are resynchronized.

See Also

For information on `ldif2db`, refer to the *Netscape Directory Server Administrator's Guide*.

dsypxfr

Synopsis

The command syntax for `dsypxfr` is:

```
/opt/SUNWconn/ldap/sbin/dsypxfr [-f ] [-C tid prot servname] [-d
domainname] [-h host] [-s domainname] [-t timeout] table
```

dsypxfr

```
/opt/SUNWconn/ldap/sbin/dsypxfr_1perday.sh  
/opt/SUNWconn/ldap/sbin/dsypxfr_1perhour.sh  
/opt/SUNWconn/ldap/sbin/dsypxfr_2perday.sh
```

Description

The `dsypxfr` command synchronizes the NIS maps between master and slave servers. It calls the `ypxfrd` daemon on the master to update the local database, and to transfer the specified NIS maps to the local host.

The `ypxfrd` must be running on the master server.

The `dsypxfr_1perday.sh`, `dsypxfr_1perhour.sh` and `dsypxfr_2perday.sh` scripts can be used to build sorted crontab entries. These scripts include several commands to update several maps together.

The `dsypxfr_1perday.sh` script updates the following maps:

- `group.byname`
- `group.bygid`
- `protocols.byname`
- `protocols.bynumber`
- `networks.byname`
- `networks.byaddr`
- `services.byname`
- `ypservers`

The `dsypxfr_1perhour.sh` script updates the following maps:

- `passwd.byname`
- `passwd.byuid`

The `dsypxfr_2perday.sh` script updates the following maps:

- `hosts.byname`
- `hosts.byaddr`
- `ethers.byaddr`
- `ethers.byname`
- `netgroup`

- netgroup.byuser
- netgroup.byhost
- mail.aliases

Options

-f

Force the transfer to occur even if the version at the master is not more recent than the local version.

-C *tid prot servname*

Used internally by `dsypserv` to pass callback information. Do not set this option.

-d *domainname*

Specify a domain other than the default domain.

-h *host*

Get the map from the specified host even if a different host name is specified in the map itself. If *host* is not specified, `dsypxfr` gets the name of the master server from the NIS service.

-s *domainname*

Specify a source domain from which to transfer a map that should be the same across domains.

-t *timeout*

Time limit for the map transfer in seconds. The default value is 25.

mapname

Name of the map to be updated.

See Also

See the man page for `ypxfrd(1M)`.

nis.at.conf

Synopsis

The location of the `nis.at.conf` file:

```
/opt/SUNWconn/ldap/default/schema/nis.at.conf
```

Description

The `nis.at.conf` file contains schema information used for synchronizing the NIS service with the Netscape Directory Server. It contains a list of LDAP attributes, described in LDIF format. The attributes in this file are not specified in RFC 2307 *An Approach for Using LDAP as a Network Information Service*, but are required to use the Netscape Directory Server as an NIS server.

For information on LDIF, refer to *Netscape Directory Server Administrator's Guide*.

Attributes in the `nis.at.conf` file are defined by:

- An OID identifying the attribute
- A name identifying the attribute
- An OID representing the attribute syntax
- If the attribute is single-valued, the keyword `SINGLE-VALUE`

For example, the line describing the `sunNisDomain` attribute is:

```
1.3.6.1.4.1.42.2.27.1.1.2 NAME 'sunNisDomain' SYNTAX  
'1.3.6.1.4.1.1466.115.121.1.26' SINGLE-VALUE
```

See Also

See “`nis.oc.conf`” on page 136.

nīs.conf

Synopsis

The location of the `nīs.conf` file is:

```
/opt/SUNWconn/ldap/default/nīs.conf
```

Description

The `nīs.conf` file contains configuration information for the NIS service. It contains the following parameters:

`rootdn`

The DN of the NIS administrator. You are prompted to provide this DN when you run the `dsypinstall` initialization script.

`rootpw`

The password of the NIS administrator. You are prompted to provide this password when you run the `dsypinstall` initialization script.

`ldapport`

The port number on which the directory server daemon `slapd-<serverID>` listens for LDAP traffic. By default it is port number 389.

`ldappath`

The path to the installation directory of the directory server. By default, the Netscape Directory Server is installed under `/usr/netscape/server4`.

More configuration parameters for the NIS service are stored in the `nīs.mapping` file.

See Also

See “`nīs.mapping`” on page 134.

nis.mapping

Synopsis

The location of the `nis.mapping` file is:

```
/etc/opt/SUNWconn/ldap/current/mapping/nis.mapping
```

Description

The `nis.mapping` file contains:

- Configuration parameters for the NIS service
- NIS-to-LDAP mapping information

Configuration Parameters

The configuration information stored in the `nis.mapping` file is at the beginning of the file, under the section entitled Configuration Variables.

The following configuration variables are defined:

`DOMAIN_NAME`

Specifies the NIS domain managed by the server.

`NAMING_CONTEXT`

When this variable is defined, it specifies the directory tree suffix under which the NIS subtree is created.

If this variable is not defined, the directory tree suffix is derived from the domain name supplied when running the `dsypinstall` script. By default the directory tree suffix is generated with `dc` (domain component) attributes. For example, with `DOMAIN_NAME=france.airius.com`, the directory tree suffix created by default is `dc=france,ds=airius,dc=com`.

The NIS subtree shown in “NIS Files/LDAP Subtrees” on page 71 is created under this subtree.

`ADMIN_SUFFIX`

The distinguished name of the subtree that will hold NIS administrative entries. These entries are maintained automatically by the server.

DBM_DIRECTORY

Specifies the directory where the NIS binary maps are generated.

AUTOMATIC_PUSH

When NIS entries are modified in the LDAP directory, specifies to automatically push modifications to slave NIS servers. This variable is used only in the context of standard NIS replication (using `dsyppush`), not in the context of LDAP replication.

The possible values for this variable are **enabled** or **disabled**. The default setting is **disabled**.

AUTOMATIC_PUSH_DELAY

Specifies the delay for pushing modifications to slaves in minutes. When this variable is defined, the `AUTOMATIC_PUSH` variable must be **enabled**.

Mapping Information

The `nis.mapping` file defines the mapping of entries in NIS files to LDAP attributes. This `nis.mapping` file is divided into sections that contain mapping information for each NIS table. This mapping information is used by the NIS initialization process, `dsypinstall`, to import information from the NIS source files into the LDAP directory. It is specifically used by the following components:

- The `dsimport` utility, to extract information from NIS files and convert it to LDAP directory entries.
- The `dsexport` utility, to extract information from LDAP entries and restore the NIS files in their original format.
- The `dsypxferd` daemon, to check that the tables specified in synchronization requests are supported by the mapping file.

The mapping definitions in the `nis.mapping` file specify the following:

- The object class and attributes used to create an entry in the directory
- The subtree under which the entries are created
- Case-sensitivity of the directory server for searches on the entries

`nis.oc.conf`

The mapping of information for each table is described in “NIS File Entries/LDAP Entries” on page 72. The syntax and semantics of the mapping definitions are described in Appendix A, “Mapping Syntax and Semantics.”

See Also

See “`nis.conf`” on page 133.

`nis.oc.conf`

Synopsis

The location of the `nis.oc.conf` file is:

```
/opt/SUNWconn/ldap/default/schema/nis.oc.conf
```

Description

The `nis.oc.conf` file contains schema information used for synchronizing the NIS service with the Netscape Directory Server. It contains a list of LDAP object classes, described in LDIF format. The object classes in this file are not specified in RFC 2307 *An Approach for Using LDAP as a Network Information Service*, but are required to use the Netscape Directory Server as an NIS server.

For information on LDIF, refer to *Netscape Directory Server Administrator's Guide*.

Object classes in the `nis.oc.conf` file are defined by:

- An OID identifying the object class
- A name identifying the object class
- A list of mandatory attributes introduced by the keyword **MUST**
- A list of optional attributes introduced by the keyword **MAY**

For example, the line describing the `nisMailAlias` object class is:

nis.oc.conf

```
1.3.6.1.4.1.42.2.27.1.2.5 NAME 'nisMailAlias' SUP 'top' MUST ( cn ) MAY  
( rfc822mailMember )
```

See Also

See “nis.at.conf” on page 132.

nis.oc.conf



Mapping Syntax and Semantics

This chapter describes the mapping syntax and semantics used in the `nis.mapping` file and the `radius.mapping` file. The mapping syntax and semantics are designed to provide maximum flexibility so that you can easily:

- Import information from any text file into the directory
- Adapt or create the mapping for a proprietary table in your NIS environment

If this involves modifying or creating an object class with the attributes that you need, refer to the *Netscape Directory Server Administrator's Guide* for instructions. For information on the default NIS schema, refer to “NIS Schema” on page 82.

File Structure

A mapping file is made up of a number of sections that conform to the following pattern:

```
Front-end name
  Common
  Table
    Common
    Dynamic
    Export
```

```
        Extract
        Condense
        Build

    Import
        Extract
        Condense
        Build

    ...
```

The content and meaning of each section is described in “Mapping Semantics” on page 140. The syntactic rules for each section are described in “Mapping Syntax” on page 145.

Mapping Semantics

This section describes the meaning of the information held in each portion of a mapping file.

Front-end name indicates the name of the service. All the information that follows that name describes the mapping of service-specific information to LDAP object classes and attributes.

The first *Common* section immediately following the front-end name gives configuration information that applies to the front-end or service. It contains mandatory configuration variables that are required in the translation process, and optional configuration variables that are stored in the same file for convenience.

The *Table* section provides a mapping definition for a particular type of information. The mapping definition determines the object class of all entries created using that particular definition. Each table definition is composed of the following sections:

- Common
- Dynamic (mandatory)
- Export
- Import

The Dynamic section is the only one that is mandatory. Without it, neither import nor export operations work. The other sections can be omitted if you do not need them. For instance, if you never intend to export information from the directory, you do not need to create an Export section.

Each section contains keywords and definitions used in the import or export process. Table A.1 provides a list of mapping keywords, the sections in which they can occur, and their purpose.

In any section, you can create variables or *tokens*, that is, private definitions, by using the following format:

```
tokenT=token definition
```

Your private definitions can use the syntax and functions described in “Condense” on page 147.

Table A.1 Summary of Mapping File Keywords

Section	Keyword	Mandatory/Optional	Purpose
Common	BASE_DN	Mandatory, but can be specified in the Dynamic section	Specifies a subtree. See “BASE_DN” on page 142.
	MAP_NAME	Mandatory for an NIS table definition	Indicates the name of the NIS table corresponding to the table definition. See “MAP_NAME” on page 143.
	PRIVATE_OBJECTCLASS ES	Mandatory when object class is not unique	Used for updates on entries created from several table definitions. See “PRIVATE_OBJECTCLASSES” on page 143.
Dynamic	ALL_FILTER	Mandatory	Defines a filter for identifying all entries created using the table definition. See “ALL_FILTER” on page 144.
	DC_NAMING	Optional	Defines the mechanism for converting a domain name to an LDAP dc name structure. See “DC_NAMING” on page 144.
	LINE	Mandatory	Defines decomposition of input information. See “LINE” on page 143.

Section	Keyword	Mandatory/Optional	Purpose
	MATCH_FILTER	Mandatory	Defines a filter for identifying a particular entry created using the table definition. See “MATCH_FILTER” on page 144.
Export/Build	LINE	Mandatory if the Export section exists	In export file, defines format of line composed of LDAP attributes. See “LINE” on page 143.
	NIS_KEY	Mandatory for NIS	Identifies NIS key in export file.
	NIS_VALUE	Mandatory for NIS	Identifies NIS value in export file.
Import/Extract	LINE	Mandatory if the Import section exists	Defines decomposition of input information. See “LINE” on page 143.

Common Section

The Common section contains definitions of variables that apply to all the entries created using that table definition but not to the entire service or front-end. For example, the Common section typically contains the subtree under which the entries are created. The subtree is specified using the `BASE_DN` keyword.

`BASE_DN`

The `BASE_DN` keyword specifies the subtree under which the entries are to be created. The `dsimport` utility looks for this parameter in several places, in the following order:

- Command line of `dsimport`, option `-V`
- Dynamic section
- Common section for the Table
- Common section for the Front-End (at the beginning of the mapping file)

MAP_NAME

The **MAP_NAME** keyword specifies the name of the NIS map corresponding to the table definition. This keyword is used to create administrative entries for the NIS service. The directory server maintains these entries automatically.

This keyword is used also to create the subtree for the NIS entries that are created by using the generic mapping definition.

The **MAP_NAME** keyword is specific to the NIS service.

PRIVATE_OBJECTCLASSES

The **PRIVATE_OBJECTCLASSES** keyword specifies an object class when the object class and attributes derived from a table definition do not make up a complete entry. This keyword is necessary for maintaining directory entries that are created from several table definitions. This can be the case when several table definitions each create an auxiliary object class and its associated attributes.

For example, in the NIS environment, network hosts can have entries in at least three files: `/etc/bootparams`, `/etc/ethers`, `/etc/hosts`. However, each host has just one entry in the LDAP directory, with the three auxiliary object classes `bootableDevice`, `ieee802Device`, and `ipHost`. If the entry for the host is deleted in one of these files, the corresponding entry in the LDAP directory must not be deleted but simply updated by removing the appropriate auxiliary object class, and any attributes specific to that object class.

Dynamic Section

The Dynamic section contains equations that make it possible to dynamically build the filters required to locate relevant information.

LINE

The **LINE** keyword is necessary to define how the input information must be dynamically decomposed to provide the elements required in the **MATCH_FILTER** and **ALL_FILTER** definitions.

The syntax of the **LINE** keyword is given in “Extract” on page 146.

`MATCH_FILTER`

The `MATCH_FILTER` keyword specifies a filter that is used by the `dsimport` utility to check whether an entry already exists in the database before creating it. If it exists, the `dsimport` utility will check whether it needs to be modified.

The `MATCH_FILTER` keyword is also used by the directory server to respond to commands such as `ypmatch`.

`ALL_FILTER`

The `ALL_FILTER` keyword specifies a filter that is used by the `dsexport` command to regenerate the file from which the directory entries were originally created. This filter is necessary even if you do not intend to export information from the directory to regenerate the source file for that information.

The `ALL_FILTER` keyword is used by the directory server to retrieve from the directory all entries that belong to a given NIS table.

The `ALL_FILTER` keyword is also used by the directory server to respond to commands such as `ypcat`.

`DC_NAMING`

The `DC_NAMING` keyword defines the mechanism applied to convert a domain name of the form `airius.com` to an LDAP data store suffix or subtree of the form `dc=airius`, `dc=com`. This is useful if the naming structure that you use in your directory is a domain component (dc) structure.

Export Section

The Export section provides the method for regenerating a source file from LDAP directory entries. This section is optional. When it exists, it must contain the keyword `LINE`. The `LINE` keyword in the Export section must reflect the format of a line in the original source file.

The Export section contains the following subsections:

- **Condense**
This optional subsection contains variable definitions that can be used in the Build subsection to build the parameters required to generate `LINE`.
- **Build**

Contains at least the output `LINE` definition.

In the `nis.mapping` file, the `Build` subsection defines the rules for constructing an NIS key/NIS value pair; it also defines the rules for generating the line in the NIS file corresponding to the LDAP directory entry.

Import Section

The `Import` section provides the method for translating a line in an input file into an LDAP directory entry. This section must contain a `LINE` keyword that defines how a line in the input file can be decomposed into elements that can be described by LDAP attributes. It must also contain the list of LDAP attributes that are created from a line in the input file.

The `Import` section contains the following subsections:

- `Extract`
Contains the `LINE` definition with the notation described in “`Extract`” on page 146.
- `Condense`
Contains variable definitions that can be used in the `Build` subsection to generate LDAP attributes and attribute values.
- `Build`
Provides a list of LDAP attributes, including the object class, and defines the rules for constructing the value of each LDAP attribute from the variables in the `Condense` section or from the parameters in the `Extract` section

In the `nis.mapping` file, the `LINE` definition in the `Extract` subsection specifies the rules for analyzing a line in an NIS source file into smaller units of information called NIS tokens.

Mapping Syntax

This section describes the syntax of the variables or tokens that you can create in each section of a table definition. The mapping syntax is described using examples from the `nis.mapping` file.

Common

The variables defined in the Common section other than the keywords listed in Table A.1 must follow this syntax:

```
variable-name=value
```

Variables defined in the Common section contain static configuration information.

Dynamic

The variables defined in the Dynamic section other than the keywords listed in Table A.1 follow the same syntax as the variables defined in the Common section. However, their values are supplied in the input to the utility (such as `dsimport` or `dsexport`) that uses the mapping file during its execution.

Extract

The variables defined in the Extract section define the rules for decomposing input information into smaller units of information, called tokens, that can be directly mapped onto LDAP attributes, or that require simple processing in order to be mapped onto LDAP attributes.

The syntax of a variable that defines a decomposition into tokens is:

```
VARIABLE => $element1 separator $element2 [separator $elementn... || ...]
```

The separator between tokens is the separator expected in the input information. It could be white space, a comma, a colon or any other character. However, one space in the line definition will match any number of spaces or tabs in the actual input information. You can specify several alternatives for the decomposition, by using two pipe symbols (`||`) to introduce each alternative rule.

The conversion process examines the rules in the order in which they are specified, and applies the first one that matches the information it was given in input.

For example, in `nis.mapping`, the following definition extracts tokens from a line in the `bootparams` file:

```
LINE =>$ipHostNameT $parametersT
```

The `hosts` file provides a slightly more complex example:

```
LINE =>$dummy $ipHostNumberT $ipHostNameT
$allIpHostAliasesT*#$descriptionT* || $dummy $ipHostNumberT
$ipHostNameT $allIpHostAliasesT || $dummy $ipHostNumberT $ipHostNameT
```

In these examples, the tokens `parametersT` and `allIpHostAliasesT` require further processing before they can be mapped onto LDAP attributes. The processing required is defined in the `Condense` section.

Condense

The `Condense` section contains variables that define operations on tokens resulting from the `Extract` section, or any previously defined variable in the table definition. It simplifies the attribute value definitions given in the `Build` section.

Variables defined in the `Condense` section can contain:

- A token specified in the same table section
- A configuration variable specified in the `Common` section for the same NIS table, or in the `Common` section that applies to all NIS tables
- A constant expression
- A function: `exclude`, `getrdn`, `split`, `instances2string`, `string2instances`, or `trim`. A variable can contain just one function. If you want to use several functions on the same information, you must create intermediate variables.

Variables in the `Condense` section can be made up of several alternative rules. The conversion process applies the first rule that matches the input information. The rules must be separated by two pipe symbols, and must all be part of the same expression. For example, the following expression is permitted:

```
fifi=$parameter1 - $parameter2 || $parameter1 || juju
```

whereas, the following expression is not:

```
fifi=$parameter1 - $parameter2
fifi=$parameter1
fifi=juju
```

You can define any number of variables in the Condense section. The order in which they are listed is important if you create dependencies between them. For instance, you can have:

```
fifi=$parameter1 - $parameter2 || $parameter1 || juju
riri=$fifi - $parameterA
loulou=$fifi - $parameterB
```

split Function

The syntax of the `split` function is as follows:

```
variableA=split(string, "separator", "add_prefix", "add_suffix", order)
```

Where:

variableA identifies the variable

string identifies the unit of information, variable or parameter, to which the operation applies

separator indicates where to split the information. This value must be specified between quotes because it could contain a space.

add_prefix specifies a prefix to add to each item resulting from the split. This value must be specified between quotes because it could contain a space.

add_suffix specifies a suffix to add to each item resulting from the split. This value must be specified between quotes because it could contain a space.

order specifies the order in which the items resulting from the split are to be presented. The possible values for this parameter are **left2right** or **right2left**.

For example, in the `nis.mapping` file, the following variable definition is used to split an NIS domain name into a sequence of LDAP domain component attributes:

```
DC_NAMING=split($DOMAIN_NAME, ".", "dc=", "", left2right)
```

If the domain name specified is `eng.europe.airius.com`, the resulting expression is:

```
dc=eng, dc=europe, dc=airius, dc=com.
```

string2instances Function

The `string2instances` function breaks down a specified string into instances. The syntax for this operation is:

```
variableA=string2instances("string", "separator")
```

Where:

variableA identifies the variable

string identifies the unit of information, variable or parameter, to which the operation applies. This value must be specified between quotes because it could contain a space.

separator indicates where to split the information. This value must be specified between quotes because it could contain a space.

For example, in `nis.mapping`, the following definition in the Condense section of the `bootparams` file breaks down a string of parameters into separate instances:

```
bootParameterT=string2instances($parametersT, " ")
```

The `string2instances` function is also used to specify the inheritance tree for an object class. For example, if the object class of an entry created using a particular mapping definition is `organizationalPerson`, the Condense section of the mapping definition must contain the line:

```
objectClassT=string2instances("top person organizationalPerson", " ")
```

instances2string **Function**

The `instances2string` function combines several instances into a single string. The syntax for this operation is:

```
variableA=instances2string(instance, "separator")
```

Where:

variableA identifies the variable

instance is a discrete element of *variableA*

separator marks the separation between the elements of the variable. This value must be specified between quotes because it could be a space.

For example, you could use the following variable to find the list of names and alias names for a given machine:

```
NameList=instances2string($cn, " ")
```

If the `cn` attribute has the values `camembert`, `Cam`, `Bertie`, the resulting string would be:

```
camembert Cam Bertie
```

`trim` **Function**

The `trim` function removes any unnecessary white space surrounding a parameter. The syntax for the `trim` operation is:

```
variableA=trim(parameter)
```

Where:

variableA identifies the variable

parameter is the item from which white space must be removed

For example, if you decompose an alias list into its constituent members, you could define the following variables:

```
aliasMember=string2instances($aliasList, ",")  
trimAliasMember=trim($aliasMember)
```

Each `aliasMember` parameter resulting from the `string2instances` operation is processed to remove any white spaces.

`getrdn` **Function**

The `getrdn` function returns the naming attribute of an entry, that is the attribute used in the entry's RDN. The syntax for the `getrdn` operation is:

```
variableA=getrdn()
```

Note The `getrdn` function can only be used in variables in the Condense section.

For example, the `cn` attribute of a machine has the values `camembert`, `Cam`, `Bertie`, but the actual system name of the machine, used in the RDN is `camembert`. For example, you could create the following variable:

```
HostName=getrdn()
```

The `getrdn` function returns the name `camembert`.

Note The `getrdn` function is case-sensitive.

`exclude` **Function**

The `exclude` function removes a value from a list or a string. The syntax for this operation is:

```
variableA=exclude(string, exclude-value, "separator")
```

Where:

variableA identifies the variable

string identifies the list or string

exclude-value is the value to exclude

separator marks the separation between the elements of the list or string. This value must be specified between quotes because it could be a space.

For example, to obtain the list of aliases for a machine, you need to exclude the canonical name from the list of names. You could create the following variables:

```
NameList=instances2string($cn, " ")
HostName=getrdn( )
HostAliases=exclude(NameList, HostName, " ")
```

In `nis.mapping`, the Condense section of the `hosts` mapping definition contains:

```
ipHostAliasesLineT=exclude($allIpHostAliasesT,$ipHostNameT, " ")
```

This definition excludes the `ipHostName` from the list of alias names for the host.

Build

The Build subsection contains a list of LDAP attributes and the definitions of their values. It must contain at least all mandatory attributes for an object class, and the DN. If the DN definition is missing from the Build section, the entries cannot be created in the directory.

Note You do not need to specify a DN definition in the Build sections of the `radius.mapping` file because this file is not used to import entries into the directory.

Attribute value definitions can be made up of:

- A variable or keyword specified in any of the sections of the table definition
- A configuration variable specified in the Common section that applies to all the tables in the Front-End section
- A constant expression
- A concatenation of any of the above

Mapping Syntax

The syntax of an LDAP attribute and its associated value definition in the Build section is as follows:

LDAPAttribute=attributeValueDefinition

For example, if you wanted to create an entry for a mail alias, and use the LDAP attribute `rfc822mailMember` to store the names of alias members, your mapping would contain the following definitions:

Condense:

```
aliasMember=string2instances($aliasList, ",")
trimAliasMember=trim($aliasMember)
```

...

Build:

```
rfc822mailMember=$trimAliasMember
```


Index

A

- access control rules
 - Deja 41
- ADMIN_SUFFIX variable 134
- aliases file 74
 - NIS-to-LDAP mapping 74
- ALL_FILTER keyword 144
- attribute
 - assigning a value 51
 - automountInformation 88
 - bootFile 89
 - bootParameter 89
 - commonName 89
 - deleting a value 52
 - description 89
 - gecos 89
 - gidNumber 89
 - ipHostNumber 90
 - ipNetmaskNumber 90
 - ipNetworkNumber 90
 - ipProtocolNumber 90
 - locality 90
 - loginShell 91
 - macAddress 91
 - manager 91
 - memberNisNetGroup 91
 - modifying a value 52
 - nisMapEntry 91
 - nisMapName 91
 - nisNetgroupTriple 92
 - nisNetIdGroup 92
 - nisNetIdHost 92
 - nisNetIdUser 92
 - oncRpcNumber 92
 - rfc822MailMember 92
 - seeAlso 93
 - selecting 50

- serialNumber 93
- shadowExpire 93
- shadowFlag 93
- shadowInactive 93
- shadowLastChange 93
- shadowMax 94
- shadowMin 94
- shadowWarning 94
- sunNisDbmCache 94
- sunNisDnsForwarding 94
- sunNisDomain 95
- SunNisInputFile 95
- SunNisKey 95
- sunNisMapFullName 95
- sunNisMapState 96
- sunNisMaster 96
- sunNisOutputName 96
- sunNisSecurityMode 96
- uid 97
- uidNumber 96
- userid 97
- userPassword 97

- AUTOMATIC_PUSH variable 135
- AUTOMATIC_PUSH_DELAY variable 135
- automount object class 83
- automountInformation attribute 88

B

- BASE_DN keyword 142
- bootableDevice object class 83
- bootFile attribute 89
- bootParameter attribute 89
- bootparams file 76
 - NIS-to-LDAP mapping 76

C

- cancel
 - create operation 53
- cn attribute
 - see commonName 89
- command
 - dejasync 40, 109
 - dsexport 110
 - dsimport 114
 - dsmakedbm 22
 - dsyp 23, 118
 - dsypaddmap 23, 119
 - dsypdelmap 23, 121
 - dsypinit 22, 122
 - dsypsync 23, 129
 - dsypxfr 22, 130
 - ypcat 144
 - ypmatch 144
- commonName attribute 89
- configuration
 - NIS service 31
- connection properties 44
- conventions, in this book 14
- copy operation 54
- copying an entry
 - from the View window 47
- creating an entry 47
 - adding an object class 50
 - removing an object class 50
 - selecting attributes 50
 - selecting object classes 50
- crypt syntax 65, 105
- cut operation 54

D

- daemon
 - dsyppasswd 22
 - dsyppasswdd 127
 - dsypserv 21
- DBM_DIRECTORY variable 135

- DC_NAMING keyword 144
- Deja 38, 44, 45, 50, 56
 - access control rules 41
 - connecting to another server 45
 - connection properties 44
 - copying an entry 54
 - creating an entry 47
 - cutting an entry 54
 - deleting an entry 53
 - display options 43
 - general parameters 61
 - host name 44
 - logging in 41
 - new window, opening 44
 - pasting an entry 55
 - port number 44
 - properties 43
 - reconnecting to server 45
 - renaming an entry 56
 - restoring an entry 56
 - search results 59
 - searching for an entry 58
 - selecting attributes 50
 - starting 40
 - toolbar icons 38
 - user properties 43
 - View window 47
 - viewing an entry 46
- Deja.properties file
 - general parameters 61
 - location 99
- dejasync command 40, 109
 - running 40
 - syntax 108
- delete operation 53
- description attribute 89
- DOMAIN_NAME variable 134
- dsexport command
 - syntax 110
- dsimport command
 - syntax 114
- dsmakedbm command 22
- dsyp command 23, 118

- syntax 118
- dsypaddmap command 23, 119
 - syntax 119
- dsypdelmap command 23, 121
 - syntax 120
- dsypinit command 22, 122
 - syntax 122
- dsypinstall command
 - syntax 125
- dsypinstall script 27, 125
- dsyppasswd daemon
 - syntax 126
- dsyppasswdd daemon 22, 127
- dsypserv daemon 21
- dsypsync command 23, 129
 - syntax 128
- dsypxfr command 22, 130
 - syntax 129

E

- entry 46
 - copying 54
 - cutting 54
 - deleting 53
 - highlighting 47
 - naming 49
 - pasting 55
- entry modifying
 - assigning attribute values 51
 - deleting attribute values 52
 - modifying attribute values 52
- entry renaming 56
- environment variable
 - JAVA_HOME 41
- ethers file 77
 - NIS-to-LDAP mapping 77
- exclude function 150

F

- file
 - Deja.properties 99
 - nis.at.conf 132
 - nis.conf 133
 - nis.mapping 134
 - nis.oc.conf 136
- fonts, in this book 14

G

- gecos attribute 89
- generic 74
- generic mapping 74
- getrdn function 150
 - mapping 150
- gidNumber attribute 89
- group file 78
 - NIS-to-LDAP mapping 78

H

- host name 44
- hosts file 78
 - NIS-to-LDAP mapping 78

I

- icons 38
- ieee802Device object class 84
- instances2string function 149
- int syntax 65, 105
- ipaddr syntax 65, 105
- ipHost object class 84
- ipHostNumber attribute 90
- ipNetmaskNumber attribute 90
- ipNetwork object class 84
- ipNetworkNumber attribute 90
- ipProtocol object class 84
- ipProtocolNumber attribute 90

J

JAVA_HOME environment variable 41

K

keywords
mapping 141

L

l attribute
see locality 90
LINE keyword 143
locality attribute 90
logging in
Deja 41
loginShell attribute 91

M

macAddress attribute 91
maintenance
NIS information 33
manager attribute 91
MAP_NAME 143
MAP_NAME keyword 143
mapping 74, 76, 77, 78, 79, 80, 81, 82, 143
ALL_FILTER 144
BASE_DN 142
DC_NAMING 144
exclude function 150
instances2string function 149
keyword summary 141
LINE 143, 144, 145
MATCH_FILTER 144
NIS-to-LDAP 72
PRIVATE_OBJECTCLASSES 143
split function 148
string2instances function 148
trim function 150
mapping file
Common section 142

Condense section 147
Dynamic section 143, 146
Export section 144
Extract section 146
Import section 145
semantics 140
structure 139
syntax 145

MATCH_FILTER keyword 144
memberNisNetGroup attribute 91
modify operation 56
modifying an entry 56

N

naming
NIS entry 49
NAMING_CONTEXT variable 134
netgroup file 79
NIS-to-LDAP mapping 79
networks file 80
NIS-to-LDAP mapping 80
NIS entry
search 58
search results 59
searching 58
NIS functions
naming an entry 49
NIS information
maintenance 33
NIS service 33
configuration 31
propagating maps 34
nis.at.conf file
location 132
nis.conf file
location 133
nis.mapping file
location 134
synchronization 109
nis.oc.conf file

- location 136
- nisMailAlias object class 85
- nisMap object class 85
- nisMapEntry attribute 91
- nisMapName attribute 91
- nisNetGroup object class 85
- nisNetgroupTriple attribute 92
- nisNetId object class 85
- nisNetIdGroup attribute 92
- nisNetIdHost attribute 92
- nisNetIdUser attribute 92
- nisObject object class 86
- nisSunObject object class 86
- NIS-to-LDAP mapping 72

O

- object class
 - automount 83
 - bootableDevice 83
 - ieee802Device 84
 - ipHost 84
 - ipNetwork 84
 - ipProtocol 84
 - nisMailAlias 85
 - nisMap 85
 - nisNetGroup 85
 - nisNetId 85
 - nisObject 86
 - nisSunObject 86
 - oncRpc 86
 - posixAccount 86
 - posixGroup 87
 - selecting 50
 - shadowAccount 87
 - sunNisMap 87
 - sunNisServer 88
- oncRpc object class 86
- oncRpcNumber attribute 92

P

- passwd file 80
 - NIS-to-LDAP mapping 80
- paste operation 55
- port number 44
- posixAccount object class 86
- posixGroup object class 87
- PRIVATE_OBJECTCLASSES keyword 143
- propagation
 - NIS maps 34
- properties
 - Deja 43
- protocols file 81
 - NIS-to-LDAP mapping 81

R

- radius.mapping file
 - synchronization 109
- reconnecting Deja 45
- refresh
 - Deja 45
- refreshing display 45
- rename operation 56
- replication
 - NIS maps 34
- restore operation 56
- rfc822MailMember attribute 92
- rpc file 81
 - NIS-to-LDAP mapping 81

S

- script
 - dsypinstall 27, 125
- search operation 58
 - NIS entry 58
 - search results 59
- seeAlso attribute 93
- selecting an object class 50

- serialNumber attribute 93
- shadowAccount object class 87
- shadowExpire attribute 93
- shadowFlag attribute 93
- shadowInactive attribute 93
- shadowLastChange attribute 93
- shadowMax attribute 94
- shadowMin attribute 94
- shadowWarning attribute 94
- special key
 - YP_INPUT_FILE 95
 - YP_OUTPUT_FILE 96
- split function 148
- string syntax 65, 105
- string2instances function 148
- styles, in this book 14
- sunNisDbmCache attribute 94
- sunNisDnsForwarding attribute 94
- sunNisDomain attribute 95
- sunNisInputFile attribute 95
- sunNisKey attribute 95
- sunNisMap object class 87
- sunNisMapFullName attribute 95
- sunNisMapState attribute 96
- sunNisMaster attribute 96
- sunNisOutputName attribute 96
- sunNisSecurityMode attribute 96
- sunNisServer object class 88
- synchronization
 - nis.mapping file 109
 - radius.mapping file 109
- syntax
 - crypt 65, 105
 - int 65, 105
 - ipaddr 65, 105
 - string 65, 105

T

- terms, in this book 14
- trim function 150

U

- uid attribute 97
- uidNumber attribute 96
- updating maps 33
- updating NIS maps 33
- user profile
 - selecting 44
- user properties 43
- userid attribute 97
- userPassword attribute 97

V

- variable
 - ADMIN_SUFFIX 134
 - AUTOMATIC_PUSH 135
 - AUTOMATIC_PUSH_DELAY 135
 - DBM_DIRECTORY 135
 - DOMAIN_NAME 134
 - NAMING_CONTEXT 134
- View window
 - closing 47
 - copy entry 47
 - displaying 47
 - highlighting an entry 47
- viewing 46
- viewing an entry 46

W

- window, closing 44

Y

- YP_INPUT_FILE special key 95
- YP_OUTPUT_FILE special key 96

- ypcat command 144
- ypmatch command 144
- ypservers file 82
 - NIS-to-LDAP mapping 82