

RADIUS 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: 806-4252-01



Recycled and Recyclable Paper

Contents

Introduction	13
RADIUS Overview	13
Prerequisite Reading	13
What Is in This Book?	13
Conventions Used in This Book	14
Chapter 1 Introducing RADIUS	15
RADIUS Authentication	15
RADIUS Accounting	16
RADIUS Architecture	17
Components Providing RADIUS Service	18
RADIUS Daemon	18
RADIUS Configuration Files	18
RADIUS Schema Objects	19
RADIUS Console	19
PAM Module	19
Chapter 2 Initializing and Configuring RADIUS	21
Initializing RADIUS	21
Configuring RADIUS	23
Starting the RADIUS Console	24
Editing RADIUS Configuration Files	24
Advanced Configuration of RADIUS Searches	27
Providing Temporary Access to Users	28
Editing the Mapping File	28
Using the RADIUS Console	29
Restricting Access through a Specified NAS	30
Editing the Mapping File	30
Using the RADIUS Console	31
Combining Temporary Access and NAS Restriction	31

Managing Virtual Domains.....	32
Processing Order for RADIUS Search Parameters.....	33
Specifying a Dictionary File	34
Configuring Dynamic Accounting.....	36
Creating a Dynamic Accounting Attribute.....	38
Specifying an Accounting File for a NAS	38
ACLs on RADIUS Information	39
RADIUS Server Statistics.....	39
Information Collected	40
Displaying RADIUS Server Statistics	42
Configuring RADIUS to Use PAM.....	42
Chapter 3 Using Deja to Update RADIUS Information	45
Introduction to Deja	46
Starting Deja	48
Logging In	49
General Operations	51
Setting the Display Options	51
Setting Deja Properties.....	51
User Properties	52
Connection Properties	52
Opening a New Deja Window	53
Closing a Deja Window	53
Reconnecting Deja to the Directory Server	53
Connecting to Another Directory Server	54
Refreshing the Browser Window	54
Operations on RADIUS Entries.....	54
Viewing an Entry	55
The View Window	55
Closing a View Window	55
Copying an Entry From a View Window.....	56
Highlighting an Entry From a View Window	56
Creating a New Entry.....	56
Naming an Entry	57

Selecting Attributes.....	58
Cancel	59
Check Data and Reply Data Attributes.....	59
Deleting an Entry.....	60
Cut, Copy and Paste.....	61
Cutting an Entry	62
Copying an Entry	62
Pasting an Entry	63
Restoring an Entry	63
Modifying an Entry.....	64
Reset.....	64
Renaming an Entry	64
Searching for an Entry	66
Remote User Searches	67
Remote Access Server Search	72
Complex Searches.....	73
Search Filters	74
Search Results List.....	75
Setting Deja Properties	76
File Structure	76
File Syntax	76
Labels	77
General Properties	77
Standard LDAP Properties	79
Hiding Attributes	79
Login Parameters	79
RADIUS Properties	80
RADIUS Search Panel Definitions.....	81
RADIUS Create Panel Definitions	83
RADIUS Profiles	84
Chapter 4 RADIUS/LDAP Information Mapping.....	87
Attribute Mapping	87
Default Mapping.....	88

Extending the Default Mapping	90
RADIUS Schema	91
RADIUS Object Classes	92
nas	92
remoteUser	92
radiusServer	93
RADIUS Attributes	93
acctattrFile	94
acctAuthentic	94
acctDelayTime	94
acctInputOctet	94
acctInputPacket	95
acctOutputOctet	95
acctOutputPacket	95
acctSessionId	95
acctSessionTime	96
acctStatusType	96
acctTerminateCause	96
authCalledStationId	96
authCallingStationId	96
authFilterId	97
authHostPortNumber	97
authHostPortType	97
authLoginService	97
authNASIdentifier	97
authPortLimit	98
authPrefixName	98
authReplyMessage	98
authServiceProtocol	98
authStartMenuId	98
authState	99
authStopMenuId	99
authType	99

authSuffixName	99
authTerminationAction	100
chapPassword	100
dictionaryFile	100
dynamicIPAddressBinding	100
dynamicIPAddress	101
dynamicSessionCounter	101
dynamicSessionId	101
expirationDate	101
framedCompression	102
framedIPAddress	102
framedMTU	102
framedProtocol	102
framedRoute	102
framedRouting	103
grpCheckInfo	103
grpReplyInfo	103
idleTimeoutNumber	103
ipLoginHost	104
ipLoginPort	104
ipxNetworkNumber	104
pamServiceName	104
radiusAuthFailedAccess	105
radiusLoginExpiration	105
radiusLoginPasswd	105
radiusLoginProfile	105
radiusPppExpiration	106
radiusPppPasswd	106
radiusPppProfile	106
radiusServerFlags	106
radiusServerRealm	106
radiusSlipExpiration	107
radiusSlipPasswd	107

radiusSlipProfile	107
sessionTimeoutNumber	107
sharedKey	108
userCallbackId	108
userCallbackNumber	108
userid	108
userPassword	109
Chapter 5 Command & File Reference.....	111
acctattr	111
Synopsis	111
Description	111
See Also	112
Deja.properties	112
Synopsis	112
Description	112
File Structure	113
File Syntax.....	113
Labels	113
User Input	114
General Properties	114
Standard LDAP Properties	115
NIS Properties	117
RADIUS Properties.....	119
See Also	121
dejasync.....	121
Synopsis	121
Description	122
nis.mapping File.....	122
radius.mapping File.....	122
Options	123
See Also	123
dictionary	123
Synopsis	123

Description.....	124
dsnmpcf.....	124
Synopsis.....	124
Description.....	124
Options.....	125
See Also.....	125
dsnmprad.....	125
Synopsis.....	125
Description.....	126
Options.....	126
See Also.....	127
dsradius.....	127
Synopsis.....	127
Description.....	128
Options.....	128
See Also.....	128
dsradiusd.....	128
Synopsis.....	128
Description.....	129
Options.....	129
See Also.....	131
dsradiusd.conf.....	132
Synopsis.....	132
Description.....	132
See Also.....	133
radius.at.conf.....	133
Synopsis.....	133
Description.....	133
See Also.....	134
radius.mapping.....	134
Synopsis.....	134
Description.....	134
Configuration Parameters.....	134

Mapping Information	136
See Also	136
radius.oc.conf	136
Synopsis	136
Description	137
See Also	137
setup_rad	138
Synopsis	138
Description	138
Options	138
Appendix A Mapping Syntax and Semantics	139
File Structure	140
Mapping Semantics	140
Common Section	143
BASE_DN	143
MAP_NAME	143
PRIVATE_OBJECTCLASSES	143
Dynamic Section	144
LINE	144
MATCH_FILTER	144
ALL_FILTER	144
DC_NAMING	145
Export Section	145
Import Section	145
Mapping Syntax	146
Common	146
Dynamic	147
Extract	147
Condense	148
split Function	149
string2instances Function	149
instances2string Function	150
trim Function	150

getrdn Function.....	151
exclude Function.....	151
Build	152
Index	155

Introduction

The RADIUS Extension Guide explains how to use the Solaris™ Extensions for Netscape Directory Server 4.11 to authenticate users through the RADIUS protocol. It also contains configuration information for the RADIUS service.

RADIUS Overview

The RADIUS server provided with Solaris Extensions for Netscape Directory Server 4.11 offers an authentication service for remote users. It also collects accounting information on remote user connections.

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 are contained in the *Netscape Directory Server Installation Guide*.

What Is in This Book?

This book explains how to configure and manage the RADIUS server provided with Solaris Extensions for Netscape Directory Server 4.11. It is intended for system administrators who are familiar with installing and configuring NAS devices and RADIUS servers.

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/...
```

Introducing RADIUS

This chapter contains background information about the RADIUS protocol and about the RADIUS server provided with Solaris Extensions for Netscape Directory Server 4.11. For configuration information, refer to Chapter 2, “Initializing and Configuring RADIUS.” For information on installing the RADIUS server refer to *Solaris Extensions Installation Guide*.

This chapter contains the following sections:

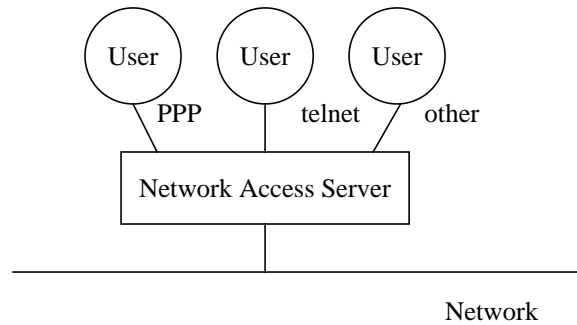
- “RADIUS Authentication” on page 15
- “RADIUS Accounting” on page 16
- “RADIUS Architecture” on page 17
- “Components Providing RADIUS Service” on page 18

RADIUS Authentication

The Remote Access Dialup User Service (RADIUS) protocol improves network security by providing a mechanism for authenticating remote users connecting to the network. It does this by carrying authentication, authorization and configuration information between a Network Access Server (NAS) and a RADIUS server.

A NAS, also known as a Remote Access Server (RAS), is a device that provides an access point to a network for remote users connecting through remote access protocols such as telnet, ftp or PPP.

Figure 1.1 Network Access Server.



The configuration shown in Figure 1.1 relies on security mechanisms provided by the connection protocol in use, for example PPP, to prevent unauthorized access to the network. Using RADIUS, you can keep a single source of authentication information in a directory and use it to authenticate remote users. The security mechanism is the same regardless of the connection protocol.

RADIUS Accounting

You can also use the RADIUS server to collect information about remote user connections. You can keep statistics on a per user basis.

A NAS can send accounting information about remote user connections to the RADIUS server. This information is logged separately for each NAS, in a log file called `detail`. The `detail` file is stored in a log directory called `/var/opt/SUNWconn/ldap/radacct/nasname`, where *nasname* is the value of the common name (cn) attribute in the directory entry for the NAS.

If the RADIUS server is unable to authenticate the NAS, accounting information is nonetheless logged, although it is marked as **unverified** in the *nasname/detail* file.

All the accounting information provided by the NAS is logged in the `detail` file.

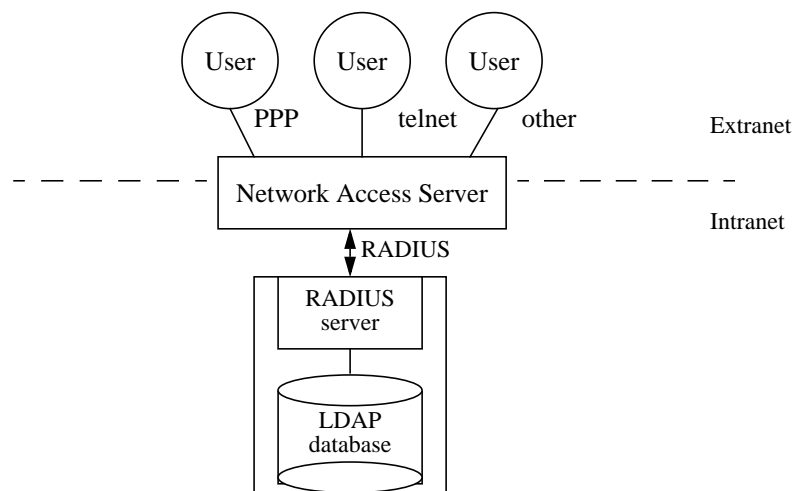
Some accounting information can also be stored dynamically in the remote user's directory entry: it is added when the user connects, and deleted when the user disconnects.

To configure the RADIUS server to log dynamic accounting information, refer to “Configuring Dynamic Accounting” on page 36.

RADIUS Architecture

When RADIUS is in use, the authentication architecture is shown in Figure 1.2.

Figure 1.2 RADIUS Authentication Architecture



A user is any entity requesting access to network resources. Each user is identified by a unique uid in the LDAP directory database.

The NAS, also referred to as the RADIUS client, is the device to which remote users connect. The client queries the RADIUS server for authentication status, user profiles and authorizations. In the directory database, each client is identified by a unique ipHostNumber. The ipHostNumber attribute, and all other attributes describing a RADIUS client are defined in the nas object class.

The RADIUS server authenticates the NAS, then checks the remote user's identity and authorization in the directory database. It returns the user's status (connection authorized or refused) and configuration information to the NAS.

If the RADIUS server is unable to authenticate the NAS, the request from the NAS is ignored. The RADIUS server does not respond, not even with a connection rejection.

Components Providing RADIUS Service

This section describes the components supplied with Solaris Extensions for Netscape Directory Server 4.11 to provide the RADIUS service.

RADIUS Daemon

The RADIUS daemon, `dsradiusd`, is the RADIUS server. It handles connection requests received from NAS devices. It connects to the directory server daemon, `ns-slapd`, to search for NAS and remote user information. It checks the information held in the directory against the information supplied in the connection request from the remote user. If the information supplied in the connection request is valid, it returns an accept-connection response that can include the connection parameters.

See “`dsradiusd`” on page 128 for more information on the RADIUS server daemon.

RADIUS Configuration Files

The configuration of the RADIUS server is mainly stored in the `dsradiusd.conf` file. However, some configuration parameters are stored in the `radius.mapping` file.

See “Configuring RADIUS” on page 23 for details on the configuration parameters.

RADIUS Schema Objects

RADIUS-specific object classes and attributes are required to support the RADIUS server, in particular to store information about NAS devices and remote users in the LDAP directory.

The RADIUS-specific LDAP attributes are mapped onto the RADIUS attributes that are actually used in the communications between the NAS and the RADIUS server. This mapping is defined in the `radius.mapping` file.

For information on RADIUS schema objects, see “RADIUS Schema” on page 91.

RADIUS Console

The RADIUS Console is a graphical tool that offers a friendly way of performing RADIUS administration and configuration tasks. It is integrated in the Netscape management framework. An icon representing the RADIUS console is listed under the Server Group subtree in the Netscape Console.

For information on starting the RADIUS Console, see “Configuring RADIUS” on page 23.

PAM Module

You can use the pluggable authentication module (PAM) to offer authentication mechanisms such as Kerberos, RSA or smart cards on top of RADIUS. The PAM module is part of the Solaris operating system.

For general information on the PAM authentication modules and API, refer to the `pam(3)` man page. For information on configuring PAM authentication modules, refer to the `pam.conf(4)` man page.

For information on configuring the RADIUS server to operate with the PAM module, refer to “Configuring RADIUS to Use PAM” on page 42.

Components Providing RADIUS Service

Initializing and Configuring RADIUS

This chapter explains how to start using RADIUS with the Netscape Directory Server. It explains the initialization process and the configuration tasks you can perform.

This chapter includes the following sections:

- “Initializing RADIUS” on page 21
- “Configuring RADIUS” on page 23
- “Advanced Configuration of RADIUS Searches” on page 27
- “RADIUS Server Statistics” on page 39
- “Configuring RADIUS to Use PAM” on page 42

Initializing RADIUS

When you initialize the RADIUS server, both authentication and accounting are enabled. There are two separate `dsradiusd` processes for authentication and accounting.

A setup script, `setup_rad`, guides you through the initialization process. The `setup_rad` script performs the following tasks:

- Registers the RADIUS console with the Netscape Console by updating the information in the configuration directory

- Configures RADIUS for use with the LDAP directory server by updating information in the user directory

Depending on how you set up your Netscape Directory Server, configuration information and user information is not necessarily stored in the same LDAP database. The `setup_rad` script takes this into account.

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

1. The full name of the machine on which you will run the RADIUS server
The format is *hostname.domainName*.
2. The LDAP URL for the configuration directory.
This is the URL to the directory that holds the configuration information. It is not necessarily the same as the directory that holds user information. The format of the URL is *ldap://hostname.domainName:portNumber/*.
3. The installation directory for the Netscape Directory Server.
4. The DN of the directory manager for the Netscape Directory Server.
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*.
5. The password of the directory manager for Netscape Directory Server.
6. The name of the administration domain managed by the server.
7. The LDAP URL for the user directory.
If you use the same directory to hold configuration information and user information, this URL is the same as the one you supplied earlier.
8. The password and DN of the directory manager for the user directory if different from the configuration directory.
9. The DN of the subtree under which the RADIUS server must perform searches for remote user and NAS authentication.

To initialize RADIUS, follow these steps:

1. Make sure that the `SUNWdsrad` and `SUNWdsutl` packages are installed. For example, type:

```
% pkginfo SUNWdsrad SUNWdsutl
```

This command should return the name and description of the packages. If you need to install the packages, refer to *Solaris Extensions Installation Guide*.

2. Make sure that the directory server daemon `ns-slapd` and the admin server daemon `ns-admin` are running. For example, type:

```
% ps -ef | grep ns-
```

The listing returned by this command should contain the following lines:

```
root  8371      1  0 16:03:13 ?          0:00 ./ns-admin -d
/usr/netscape/server4/admin-serv/config
root  8375      1  0 16:03:42 ?          0:05 ./ns-slapd -f
/usr/netscape/server4/slapd-faerie/config/slapd.conf -i /usr/nets
```

If it doesn't you must start the `ns-slapd` daemon, as explained in the *Netscape Directory Server Administrator's Guide*.

3. Run the RADIUS setup script, `setup_rad`. As root, type:

```
# /opt/SUNWconn/ldap/sbin/setup_rad
```

4. Start the Netscape Console.
5. In the left pane of the Console, browse down the tree to the RADIUS server, and double click on the icon to open the RADIUS console.
6. In the RADIUS console window, from the Task tab, select Start RADIUS Server.

Your RADIUS server is initialized and configured. If you want to change configuration parameters at any time, refer to "Configuring RADIUS" on page 23.

Configuring RADIUS

The RADIUS service can be configured in two ways:

- From the RADIUS console
- By editing the configuration files `radius.mapping`, and `dsradiusd.conf`

Starting the RADIUS Console

To start the RADIUS console:

1. Start the Netscape Console.
2. In the left pane of the Console, browse down the tree to the RADIUS server, and double click on the icon to open the RADIUS console.

The RADIUS console is displayed

To access the main configuration panel, in the RADIUS tree view on the left, select Common, then click the Configuration tab. The configuration panel contains fields for all the parameters described in “Editing RADIUS Configuration Files” on page 24.

Information on how to perform advanced configuration tasks from the RADIUS console is provided with each task description in this chapter.

Editing RADIUS Configuration Files

The RADIUS configuration is stored in the following files:

- `/etc/opt/SUNWconn/ldap/current/mapping/radius.mapping`
- `/etc/opt/SUNWconn/ldap/current/dsradiusd.conf`

Note When you modify the RADIUS configuration by editing one of these files, you must restart or refresh the RADIUS daemon. You can do this from the Tasks tab on the RADIUS console.

`radius.mapping` file

The following configuration parameters are defined in the `radius.mapping` file:

- `Max_allowed_failures`
This parameter determines blocking mode. It defines the number of permitted consecutive failures to authenticate a user based on the password provided. Any further attempt is systematically blocked, even if the connection parameters supplied are correct. The count is reset on first success. The default value for this parameter is **4**. To disable blocking mode, set this parameter to **0**.
When a user account is blocked, you must manually enable it again by setting the `authFailedAccess` attribute in the remote user’s entry to zero.

- `Dynamic`

This parameter determines whether dynamic accounting data is recorded in the LDAP directory (see “Configuring Dynamic Accounting” on page 36). It can have one of two values, **on** or **off**. By default, dynamic accounting is **off**.

- `Authentication_Port`

The authentication port number for RADIUS processes. The default port number is **1645**. Due to later standardization, the standard port number is **1812**. This port number is defined in the `radius.mapping` file but is commented out.

- `Accounting_Port`

The accounting port number for RADIUS processes. The default port number is **1646**. Due to later standardization, the standard port number is **1813**. This port number is defined in the `radius.mapping` file but is commented out.

- `Accounting_dir`

The directory where accounting information is stored. The default directory is `/var/opt/SUNWconn/ldap/radacct`. Note that there is a typographical error in this variable in the `radius.mapping` file. Do not correct it because the RADIUS server would be unable to read it.

- `Max_wait_b4_reject`

This parameter defines a client timeout on a RADIUS request. It determines the maximum time a client will wait for a response from the RADIUS server. By default it is **58** seconds.

- `Time_limit`

This parameter defines a RADIUS server timeout on LDAP search operations. It must be *strictly less than* the value of the `Max_wait_b4_reject` parameter. By default it is **50** seconds.

Note Blocking mode and dynamic accounting settings are not taken into account when the RADIUS search is performed on a referral server.

Whenever you modify the `radius.mapping` file, you must run the `dejasync` utility to copy the modifications you made to the `Deja.properties` file. Your modifications will not be reflected in `Deja` if you don't use `dejasync`.

`dsradiusd.conf` file

The following configuration parameters are defined in `dsradiusd.conf`:

- The hostname for the LDAP server
This parameter is stored in the `LDAP_Server` configuration variable. Its default value is **localhost**.
- The TCP port number that the LDAP server runs on
This parameter is stored in the `LDAP_Port` configuration variable. Its default value is **389**.
- The DN used to bind to the LDAP directory for performing searches and modifying the LDAP database
This parameter is stored in the `LDAP_Bind_dn` configuration variable. A placeholder value of **cn=radiusadmin** is provided as an example.
- The password used to bind to the LDAP directory for performing searches and modifying the LDAP database
This parameter is stored in the `LDAP_Password` configuration variable. A placeholder value of **secret** is provided as an example.
- The log level
This parameter is stored in the `LogLevel` configuration variable. You can select one of the following log levels:
 - 0 = none
 - 1 = trace
 - 2 = trace and translation
 - 3 = trace, translation, and other debug informationThe default is **0**.
- The number of kilobytes before the log file automatically changes over to the next log file.
This parameter is stored in the `LogSize` configuration variable. Its default value is **1000**.
- The log directory where `dsradius.log` is located
This parameter is stored in the `LogDir` configuration variable. Its default value is **/var/opt/SUNWconn/ldap/log**.

Advanced Configuration of RADIUS Searches

You can perform the following advanced configuration tasks:

- Provide temporary access to a remote user whose entry is not in the subtree normally searched to find users
- Associate remote users with a particular NAS, and grant access through that NAS only
- Combine the options above to provide temporary access through a specified NAS
- Manage remote users connecting from a virtual domain

These tasks are performed by modifying the DN and the search filters that the RADIUS server uses to do searches in the directory.

The subtrees searched by the RADIUS server are specified in the `radius.mapping` file by the `BaseDN` variable. The `BaseDN` variable for remote users is located in the Common section under the `USERS` table. The `BaseDN` variable for NAS devices is located in the Common section under the `RAS` table.

For example, in the Common section of the `Users` table, the `BaseDN` is similar to:

```
BaseDN= o=airius, c=us
```

The actual value of this attribute is supplied during the `setup_rad` process.

The object classes and attributes that are used in RADIUS searches are specified in the mapping file by the `FILTER` variable. The `FILTER` variable for remote users is located in the Dynamic section under the `USERS` table. The `FILTER` variable for NAS devices is located in the Dynamic section under the `RAS` table.

For example, in the Dynamic section of the `USERS` table, the `FILTER` is:

```
FILTER=( &(Objectclass=remoteUser)(uid=$UserID) )
```

In these examples, the RADIUS server searches for the `userid` passed in the request from the NAS in the subtree `o=airius, c=us` among all entries with an object class of `remoteUser`.

Note The syntax of filters is described in RFC 2254 *The String Representation of LDAP Search Filters*.

Providing Temporary Access to Users

The basic configuration for RADIUS searches on remote users is defined in the `radius.mapping` file, under the `USERS` table. The variables that define the search criteria are:

```
BaseDN= search_base  
FILTER=(&(Objectclass=remoteUser)(uid=$UserID))
```

Where:

search_base is the subtree that is searched for entries with the `remoteUser` object class.

\$UserID represents the actual `userid` passed in the request from the NAS.

Without changing your basic configuration, you can allow temporary access to a remote user whose entry is in a different subtree.

Editing the Mapping File

1. In the `USERS` table, add a `BaseDN` and `FILTER` token to the configuration with the prefix `TMP_`, and assign temporary values, using the following format:

```
TMP_BaseDN = new_search_base  
TMP_FILTER = (&(Objectclass=remoteUser)(uid=$UserID)(uid=userid))
```

where:

new_search_base is the subtree that holds the `remoteUser` entry for the person to whom you are granting temporary access. If this subtree is stored on a different server, ensure that a referral is defined between the two servers

userid is the actual `userid` of the remote user. This ensures that you grant access to that user alone, and not to all the entries with the object class `remoteUser` in the new search base

2. Restart the `dsradiusd` daemon so that the new configuration file is taken into account. As `root`, type the following commands:

```
# /opt/SUNWconn/ldap/sbin/dsradius stop  
# /opt/SUNWconn/ldap/sbin/dsradius start
```

For example, if your Base DN for remote users is `l=Paris, o=airius, c=US`, and you want to provide temporary access to the remote user Felipe Gonzalez located in Madrid, you would change the local `radius.mapping` file to include:

```
BaseDN= l=Paris, o=airius, c=us
TMP_BaseDN= l=Madrid, o=airius, c=us
FILTER=(&(Objectclass=remoteUser)(uid=$UserID))
TMP_FILTER=(&(Objectclass=remoteUser)(uid=$UserID)(uid=fgonzalez))
```

This example assumes that a referral exists between the local directory server and the directory server holding the subtree `l=Madrid, o=airius, c=US`.

If you want to temporarily extend permission to all remote users within the Airius corporation, you would use the following temporary base DN variable:

```
TMP_BaseDN= o=airius, c=us
```

This example assumes that there is a default referral from the server that holds the `l=Paris, o=airius, c=us` subtree to the server that holds the `o=airius, c=us` subtree. It also assumes that the `o=airius, c=us` subtree contains referral entries to all subtrees held on other servers.

Using the RADIUS Console

To perform the same operation from the RADIUS console:

1. In the RADIUS tree view on the left, select Users, then select the Configuration | Common tab.
2. Add a `TMP_BaseDN` variable. Do not remove the current `BaseDN` variable.
3. Select the Configuration | Dynamic tab.
4. Add a `TMP_FILTER` variable. Do not remove the current `FILTER` variable.
5. Select the Tasks tab and click Refresh.

For details about the `TMP_BaseDN` and `TMP_FILTER` variables, refer to “Editing the Mapping File” on page 28.

Restricting Access through a Specified NAS

You may want to ensure that remote users always connect to a specific NAS. For example, if you want to control communications costs, you can ensure they connect to the NAS that is geographically closest to them.

The basic configuration for RADIUS searches on NAS devices is defined in the `radius.mapping` file, under the RAS table. The variables that define the search criteria are:

```
BaseDN= search_base
FILTER= (&(Objectclass=remoteUser)(uid=$UserID))
```

Editing the Mapping File

1. In the USERS table, add a `BaseDN` and `FILTER` token to the configuration with the suffix `_nasname`, and assign temporary values.

```
BASEDN_nasname= search_base
FILTER_nasname= (&(Objectclass=remoteUser)(uid=$UserID))
```

where:

- `search_base` is the subtree that holds the directory entries for the remote users to whom you are granting access through the NAS
- `nasname` is the name of the NAS (value of the `cn` attribute in the directory entry for the NAS) through which you are granting access

For information on the order in which these lines are processed during a RADIUS search, refer to “Processing Order for RADIUS Search Parameters” on page 33.

2. Restart the `dsradiusd` daemon so that the new configuration file is taken into account. As `root`, type the following commands:

```
# /opt/SUNWconn/ldap/sbin/dsradius stop
# /opt/SUNWconn/ldap/sbin/dsradius start
```

For example, your Base DN for remote users is `l=France, o=airius, c=us`, and you have remote users located in Paris, Lyon, and Toulouse who can connect to the network through a local NAS at each site. The NAS names are `ParisNAS`, `LyonNAS`, and `ToulouseNAS`, respectively. You want to ensure that remote users always connect through the nearest NAS to save on communication costs.

You would change the `radius.mapping` file to include:

```
BasedN= l=France, o=airius, c=us
BasedN_ParisNAS= l=Paris, l=France, o=airius, c=us
BasedN_LyonNAS= l=Lyon, l=France, o=airius, c=us
BasedN_ToulouseNAS= l=Toulouse, l=France, o=airius, c=us
FILTER=(&(Objectclass=remoteUser)(uid=$UserID))
```

When the RADIUS server receives a request from ParisNAS, it checks that the remote user belongs to the naming context `l=Paris, l=France, o=airius, c=us`.

Using the RADIUS Console

To perform the same operation from the RADIUS console:

1. In the RADIUS tree view on the left, select NAS, then select the Configuration | Common tab.
2. Add a `BasedN_nasname` variable. Do not remove the current `BasedN` variable.
nasname is the name of the NAS (value of the `cn` attribute in the directory entry for the NAS) through which you are granting access.
3. Select the Configuration | Dynamic tab.
4. Add a `FILTER_nasname` variable. Do not remove the current `FILTER` variable.
nasname is the name of the NAS (value of the `cn` attribute in the directory entry for the NAS) through which you are granting access.
5. Select the Tasks tab and click Refresh.

For details about the `BasedN_nasname` and `FILTER_nasname` variables, refer to “Editing the Mapping File” on page 30.

Combining Temporary Access and NAS Restriction

You can combine temporary access permission *and* restrict access to a particular NAS by combining the `TMP_` prefix and `_nasname` suffix on the `BasedN` or the `FILTER` tokens.

For example, if you want to grant Felipe Gonzalez from Madrid remote access to the Paris NAS just for the duration of a business trip to Paris, you would modify the `radius.mapping` file to include the following lines in the Dynamic section of the USERS table:

```
BasedN= l=France, o=airius, c=us
TMP_BaseDN= l=Madrid, o=airius, c=us
FILTER=(&(Objectclass=remoteUser)(uid=$UserID))
TMP_FILTER_ParisNAS=(&(Objectclass=remoteUser)(uid=$UserID)(uid=fgonzal
ez))
```

In this example, the `_nasname` suffix is added to the temporary filter rather than to the temporary base DN. The reason is that you may want to grant other people from the Madrid office access through a different NAS from the Paris NAS. In this case, the temporary base DN remains valid, you just need to create the temporary filter with the appropriate `_nasname` suffix.

Managing Virtual Domains

You can manage remote user connections from users who belong to a virtual domain, that is, a domain that you manage on behalf of another organization.

For example, if ABC corporation decided to use ISP corporation to manage their internet mail service, ABC would be assigned a domain name such as `abc.com`, and a pool of IP addresses. ISP corporation manages user information, and remote user connections for ABC corporation. When an employee from ABC corporation connects to request remote access, the connection parameters are the user login and the user password.

For example, John Smith logs in with the following parameters:

- Login: `jsmith@abc.com`
- Password: `secret`

The RADIUS server of ISP corporation needs to separate the user ID from the domain information. This is done in the `radius.mapping` file. The beginning of the USERS table and the variables defined in the Dynamic section would be as follows:

Table: USERS

Common:

BaseDN= o=isp, c=us

Dynamic

userID=>\$myID@\$virtualDomainT || \$myID

FILTER=(&(Objectclass=remoteUser)(uid=\$myID))

In this configuration example, the `userID` variable accepts two alternative expressions so that it can handle equally well remote users who have a domain name appended to their user ID, and those who do not.

The domain name must be checked during the authentication procedure, therefore the directory entry of John Smith includes these attributes:

- uid: jsmith
- userPassword: * (protected)
- authSuffixName: @abc.com
- grpCheckInfo: userPassword, authSuffixName

The `grpCheckInfo` attribute gives the list of attributes that must be supplied in the authentication procedure. The RADIUS server checks the values provided for these attributes against the values stored in the directory database.

Processing Order for RADIUS Search Parameters

During a search, the RADIUS server handles the `BaseDN` and `FILTER` tokens in the following manner: it first performs an ordinary search, then, if this search fails, it performs a search on temporary tokens.

The ordinary search starts from the most restrictive to the most general:

1. `FILTER_nasname`
2. `FILTER`
3. `BaseDN_nasname`
4. `BaseDN`

If the ordinary search fails, the temporary search is handled in the same way:

1. `TMP_FILTER_nasname`
2. `TMP_FILTER`
3. `TMP_BaseDN_nasname`
4. `TMP_BaseDN`

Specifying a Dictionary File

The RADIUS server uses a dictionary file to convert numerical values used by the protocol to attribute names used in the `radius.mapping` file and the RADIUS log files. The dictionary file contains RADIUS attribute and numerical value pairs. A number of these attributes are defined in RFC 2138 *Remote Authentication Dial In User Service (RADIUS)*, and RFC 2139 *RADIUS Accounting*. However, NAS vendors have also defined proprietary attributes, referred to as vendor-specific attributes or VSAs.

Do not confuse the RADIUS dictionary file with the RADIUS mapping file which provides a translation between RADIUS attributes and LDAP attributes. For information on the RADIUS mapping file, refer to Chapter 4, “RADIUS/LDAP Information Mapping.”

Solaris Extensions for Netscape Directory Server 4.11 provide a default dictionary that contains the standard attribute and value definitions. It also accepts the dictionaries from the following vendors:

- Livingston
- Ascend
- Cisco
- Shiva
- Bay Networks

The dictionary files provided by vendors contain both standard and proprietary definitions. Attribute and value definitions are identified by an OID which is the actual information passed in a RADIUS transaction. Due to a lack of standardization some proprietary attributes defined by different vendors use the same OID.

The RADIUS server can support any number of dictionary files from different vendors, but you must specify which dictionary to use with a particular NAS.

To specify a dictionary file for a NAS, use the `Deja` tool to add the `dictionaryFile` attribute to the directory entry for the NAS. The value you assign to this attribute must be the filename of the dictionary that the RADIUS server must use for communications with the NAS described by the entry.

If the `dictionaryFile` attribute is not specified, the default dictionary file is used. This file, called `dictionary`, is located with all other configuration files under `/etc/opt/SUNWconn/ldap/current`.

Note If you use the dictionary provided by the NAS vendor instead of the default dictionary provided with Solaris Extensions for Netscape Directory Server 4.11, you must copy the attributes used internally by the RADIUS server from the default dictionary to the vendor-supplied dictionary. The list of attributes that you must copy is shown in the following file extract.

Table 2.1 RADIUS Server Internal Attributes

```
# Non-Protocol Attributes
# These attributes are used internally by the server
#
ATTRIBUTE   Expiration      21   date
ATTRIBUTE   Auth-Type        1000  integer
ATTRIBUTE   Menu           1001  string
ATTRIBUTE   Termination-Menu 1002  string
ATTRIBUTE   Prefix         1003  string
ATTRIBUTE   Suffix         1004  string
ATTRIBUTE   Group          1005  string
ATTRIBUTE   Crypt-Password  1006  string
ATTRIBUTE   Connect-Rate    1007  integer
#
# SUN RADIUS Attributes for LDAP Integration
#
ATTRIBUTE   Login-Profile   2000  integer
ATTRIBUTE   Login-Passwd    2001  string
ATTRIBUTE   Login-Expiration 2002  date
```

Non-Protocol Attributes

ATTRIBUTE	PPP-Profile	2010	integer
ATTRIBUTE	PPP-Passwd	2011	string
ATTRIBUTE	PPP-Expiration	2012	date
ATTRIBUTE	SLIP-Profile	2020	integer
ATTRIBUTE	SLIP-Passwd	2021	string
ATTRIBUTE	SLIP-Expiration	2022	date
ATTRIBUTE	Auth-Failed-Access	2100	integer
ATTRIBUTE	Dynamic-Session-Counter	2201	integer
ATTRIBUTE	Dynamic-SessionId	2202	string
ATTRIBUTE	Dynamic-IPAddress	2203	ipaddr
ATTRIBUTE	Dynamic-IPAddr-Binding	2204	string
ATTRIBUTE	PAM-Service-Name	2205	string

Configuring Dynamic Accounting

You can use the RADIUS server to record connection parameters dynamically in the directory entry of a remote user. To enable dynamic accounting, in the RADIUS Console, set the Dynamic Data option to On.

With dynamic accounting enabled, the following attributes are automatically added to a remote user's entry when the user connects, and removed when the user disconnects:

- Dynamic IP address:
The IP address assigned to the remote user connection
- Dynamic session ID
The accounting session ID assigned to a remote user for a given session
- Dynamic session counter
The number of concurrent open sessions
- Dynamic IP address binding
The association between the IP address and the accounting session ID for a given session

- All other RADIUS attributes listed in the accounting file

Note When a remote user has several open sessions, the dynamic session counter attribute is removed from the user's entry when the user has closed all running sessions.

A default accounting file is provided with Solaris Extensions for Netscape Directory Server 4.11, called `acctattr`. You can, if you want, create your own dynamic accounting file. The only requirement is that the file name should end with the **attr** suffix.

You must make sure that the NAS can provide the accounting parameters listed in the accounting file. This file should be located with other configuration files in `/etc/opt/SUNWconn/ldap/current`.

The dynamic accounting parameters listed in the default `acctattr` file are RADIUS attributes that can be contained in RADIUS accounting packets. The corresponding LDAP attributes are shown in Table 4.1.

The default `acctattr` file contains examples of suitable RADIUS attributes commented out. These are:

- Framed-IP-Address
- Acct-Session-Id
- NAS-Port
- NAS-Port-Type
- NAS-IP-Address

If you want to add accounting items to the list, ensure that:

1. At least one NAS can provide these items in an accounting packet.
2. There is an LDAP attribute for each RADIUS parameter that you want to record.
If there isn't, you must create the corresponding LDAP attribute, as explained in "Creating a Dynamic Accounting Attribute" on page 38.
3. The mapping between the RADIUS attribute and the LDAP attribute is defined in the `radius.mapping` file.
If it isn't, you must create it as described in "Creating a Dynamic Accounting Attribute" on page 38.

Creating a Dynamic Accounting Attribute

To create a dynamic accounting attribute, and the RADIUS/LDAP mapping definition:

1. Create an LDAP attribute for the connection parameter that you want to record.
This is a modification of the Netscape Directory Server schema. For information on how to create new attributes in the directory server schema, refer to the *Netscape Directory Server Administrator's Guide*.
2. Add the attribute to the list in the `radius.mapping` file using a text editor.
Make sure you add it in both the Import section and the Export section of the mapping file. You need to be logged in as `root` to perform this operation.
Alternatively, you can add the attribute from the RADIUS console: In the RADIUS tree view on the left, select Users, then select the Configuration | Import tab. Add the attribute in the Import section of the `radius.mapping` file. Then select the Configuration | Export tab to add the attribute to the Export section.
3. Add the attribute to the list in the accounting file using a text editor.
You need to be logged in as `root` to perform this operation.
Alternatively, you can add the attribute from the RADIUS console: In the RADIUS tree view on the left, select Users, then select the Configuration | Accounting tab. Select the accounting file that you want to modify and click Load.
4. Restart the `ns-slapd` daemon so that the new accounting attribute created in the schema is taken into account and can be recorded dynamically in remote user entries.
5. Restart the `dsradiusd` daemon so that the new `radius.mapping` file is taken into account.
To do this from the RADIUS console, from the Tasks tab, select Refresh.

Specifying an Accounting File for a NAS

To specify an accounting file for a NAS, use the Deja tool to add the `acctattrFile` attribute to the directory entry for the NAS. The value you assign to this attribute must be the filename of the dynamic accounting attribute file that the RADIUS server uses to record the dynamic accounting information received from the NAS.

If the `acctattrFile` attribute is not specified, the default `acctattr` file is used. This file is located with all other configuration files under `/etc/opt/SUNWconn/ldap/current`.

ACLs on RADIUS Information

RADIUS information in the LDAP directory is protected by a special ACI. The instruction specifies a filter for the RADIUS object classes, the list of RADIUS attributes to which the instruction applies, a name for the instruction, the permission level, and the LDAP URL used by the RADIUS server to perform searches and modifications in the LDAP directory.

The default ACI on RADIUS information is shown below.

```
# SUN Radius Attribute Control Item
#
aci:
(targetfilter="(|(objectclass=nas)(objectclass=remoteUser))")
(targetattr="*")
(version 3.0;aci "Radius User permissions"; allow(all)
userdn="ldap:///cn=radiusAdmin,o=sun.com";)
```

Note Note that the default ACI gives the RADIUS administrator all permissions on all attributes (`targetattr="*"`). You can if you want, restrict the attributes which the RADIUS administrator can access by listing them in the ACI. If you modify the default ACI, make sure that your list of attributes contains at least the `userPassword` attribute.

RADIUS Server Statistics

This section lists the information collected by the RADIUS server SNMP agent `dsnmprad`. This information can be monitored from a management platform such as Solstice™ Enterprise Manager™, Solstice Site Manager™, or Solstice Domain Manager™.

Information Collected

The information collected by the `dsnmprad` SNMP agent is defined in RFC 2619 *RADIUS Authentication Server MIB* and RFC 2621 *RADIUS Accounting Server MIB*.

The following RADIUS authentication service information is monitored:

- Server identifier
- Uptime
- Reset time
- Configuration reset
- Total access requests
- Total invalid requests
- Total duplicate access requests
- Total access accepts
- Total access rejects
- Total access challenges
- Total malformed access requests
- Total bad authenticators
- Total packets dropped
- Total unknown type
- Client entry (contains authentication information monitored for every NAS connected to the server)
 - Client Index
 - Client Address
 - Client ID
 - Access requests
 - Duplicate access requests
 - Access accepts
 - Access rejects
 - Access challenges
 - Malformed access requests
 - Bad authenticators
 - Packets dropped
 - Unknown type

The following RADIUS accounting service information is monitored:

- NAS identifier
- Uptime
- Reset time
- Configuration reset
- Total requests
- Total invalid requests
- Total duplicate requests
- Total responses
- Total malformed requests
- Total bad authenticators
- Total packets dropped
- Total no record
- Total unknown type
- Client entry (contains accounting information monitored for every NAS connected to the server)
 - Client Index
 - Client Address
 - Client ID
 - Packets dropped
 - Requests
 - Duplicate requests
 - Responses
 - Bad authenticators
 - Malformed requests
 - No record
 - Unknown type

Displaying RADIUS Server Statistics

You cannot display RADIUS server statistics in the RADIUS console. You need a management application such as Solstice Enterprise Manager, Solstice Domain Manager, or Solstice Site Manager. The files required to interoperate with these management applications are provided with Solaris Extensions for Netscape Directory Server 4.11:

- The directory `/opt/SUNWconn/ldap/snmp/snm` contains all files necessary for `dsnmprad` to report events to a Solstice Domain Manager, or Solstice Site Manager station
- The directory `/opt/SUNWconn/ldap/snmp/sem` contains all files necessary for `dsnmprad` to report events to a Solstice Enterprise Manager station.

Configuring RADIUS to Use PAM

This section describes how to enable PAM on top of the RADIUS server. The procedure given in this section is based on a sample PAM plug-in provided in the directory `/opt/SUNWconn/ldap/samples/pam`. This example PAM module can be used by the RADIUS server to authenticate, authorize and perform accounting for remote users.

1. Change directory to the `/opt/SUNWconn/ldap/samples/pam` directory.
2. Make the PAM module by typing `make` in this directory. Ignore the warning messages displayed at the end of the `make` process.
3. Copy the PAM module `pam_sample.so.1` to `/usr/lib/security/` and make sure it is owned by `root`.
4. In the RADIUS console, go to the main configuration panel and enable the use of PAM: check the PAM enabled checkbox, and specify a challenge response timeout in the Challenge/Resp Timeout field. A reasonable value is **60**.

Alternatively, you can edit the `radius.mapping` file in the directory `/etc/opt/SUNWconn/ldap/current/mapping/` to add the following lines to the Common section:

```
Pam_Authentication=on
Challenge_Response_Timeout=60
```

5. Modify the `/etc/pam.conf` configuration file to include the following lines:

```
# Radius
radius  auth  required  /usr/lib/security/pam_sample.so.1
radius  account required      /usr/lib/security/pam_sample.so.1
radius  session required      /usr/lib/security/pam_sample.so.1
radius  password required      /usr/lib/security/pam_sample.so.1
```

6. Modify the entries of remote users in the directory to include the following attributes and values:

pamServiceName: **radius**

grpCheckInfo: **pamServiceName**

The `pamServiceName` attribute specifies the name of the PAM module to use in `/etc/pam.conf`. The `grpCheckInfo` attribute usually also contains the `userPassword` attribute. If you remove `userPassword`, then only PAM authentication is used.

7. Restart the `dsradiusd` daemon so that the new `radius.mapping` file is taken into account.

To do this from the RADIUS console, from the Tasks tab, select Refresh.

Configuring RADIUS to Use PAM

Using Deja to Update RADIUS Information

This chapter explains how to use the Deja tool to add, delete and modify RADIUS information in the LDAP directory. Solaris Extensions for Netscape Directory Server 4.11 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 RADIUS 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 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 46
- “General Operations” on page 51

- “Operations on RADIUS Entries” on page 54
- “Setting Deja Properties” on page 76

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.

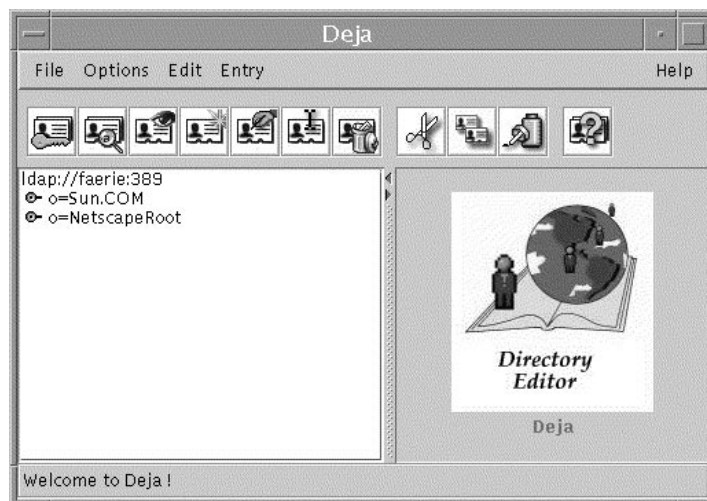













Figure 3.1 Deja Directory Editor

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
	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.
	Select an entry in the directory browser window and click this icon to copy the entry into the clipboard.
	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
	Click this icon 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 the `dejasync` utility. As root type:

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

For details on the options of the `dejasync` command, refer to “`dejasync`” on page 121. You must run `dejasync` so that Deja will take into account all the configuration options you set during the `setup_rad` 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 your entry in the User text field:
If you need to log in often, you can create a login alias in the `Deja.properties` file. See “Setting Deja Properties” on page 76 for information on creating a login alias.

If you cannot remember your full DN, you can search for it in the directory:

1. Type your user name or a substring of your name in the User text field and click the Search button in the login panel. The search can include the wildcard character `*`.
2. Double-click on your name in the Matching Usernames window.
The DN is transferred to the User text field.
3. Type your password in the Password field.
4. Select the desired profile (Standard, NIS or RADIUS) 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 when Deja is started.

When you start Deja, you can specify the host name and port number on the command line. See “Starting Deja” on page 48.

To connect to a different directory server or change the port number from within Deja see “Connecting to Another Directory Server” on page 54.

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.

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 RADIUS Entries

This section describes the read, create, modify, delete and search operations that can be performed on directory entries using Deja. Deja offers specific templates for creating, modifying and searching for RADIUS entries.

To view the RADIUS-specific panels you must change the Deja user profile to RADIUS, as explained in “User Properties” on page 52.

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 with an example entry. 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.



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 can be used to add new entries to the directory.

Figure 3.5 shows the Deja Create panel for RADIUS users.

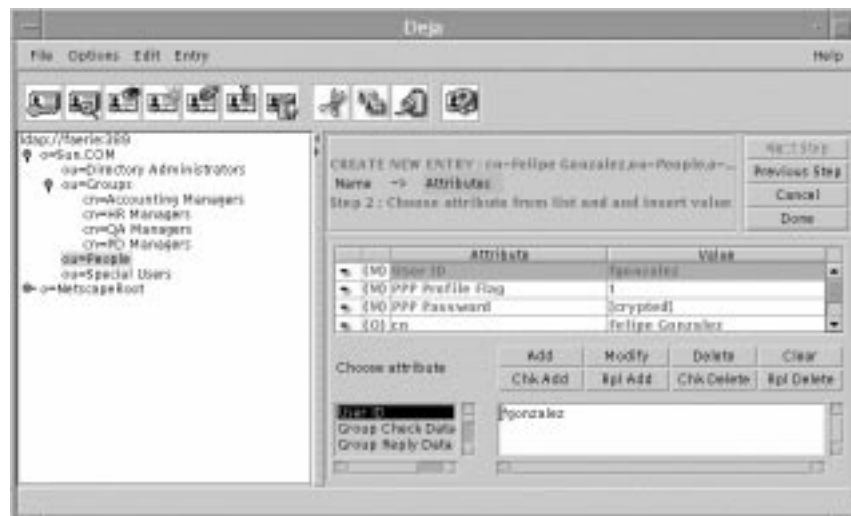


Figure 3.5 Deja Create Panel for RADIUS Users

1. Log in to Deja as a user that has write permissions to the directory.
2. Click on the Create icon or select Create from the Entry menu.

The Create panel is displayed.

There are two steps to creating a RADIUS 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 57.
- Assign attributes to the entry and name them. See “Selecting Attributes” on page 58.

3. When you have completed the entry definition, click Done.

Naming an Entry

To assign a name to an entry:

1. Select the type of entry you want to add (Remote User or Remote Access Server).
2. If you are adding a Remote User, specify the profile of the new entry (Standard, PPP, SLIP, LOGIN).

The list of RADIUS profiles available in Deja is defined in the `Deja.properties` file on the directory server. See “RADIUS Profiles” on page 84 for information on defining RADIUS user profiles.

3. Specify the parent of the entry.

By default, the Parent text field holds the distinguished name of an entry specified in the `Deja.properties` file on the directory server. To select another parent entry:

- Type the Distinguished Name of the Entry’s parent in the Parent text field.
- Alternatively, click once on the parent in the browser window to select it and 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.

4. Name the entry by selecting a naming attribute with the option button next to the Entry’s name field.

The list of available naming attributes is defined in the `Deja.properties` file on the directory server.

5. Type the value for the naming attribute of the entry in the Entry Name text field.

6. When you are satisfied with the entry name and parent, click the Next Step button to assign values to the attributes.
7. The list of attributes available for selection is different depending on the type of entry you selected in Step 1 and Step 2.
See “Selecting Attributes” on page 58 for information on selecting attributes for the entry. The attributes available for selection are defined for each object class in the schema.

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.

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.
2. Type the value for the attribute 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. For information on the Chk Add, Rpl Add, Chk Delete and Rpl Delete buttons, see “Check Data and Reply Data Attributes” on page 59.
4. To add another value for an attribute, repeat steps 1 to 3.

5. Click on Done to add the entry to the directory.
6. Double click on the entry in the browser to display all of its attributes.

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. Change the value and click Modify.

The modified value appears in the entry definition.

Cancel

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

Check Data and Reply Data Attributes

The RADIUS attribute selection window features four special buttons:

- *Chk Add*

Select an attribute and type a value for it in the text window. If you click the Chk Add button, the value is added to the entry definition, and the name of the attribute is added to the Radius Check Data optional attribute which matches the grpCheckInfo attribute in the directory schema for RADIUS.

For example, if you select the User ID attribute from the Choose Attribute list and type the value **charles** in the text window, when you click on Chk Add, the value **charles** is added to the User ID attribute, and the uid attribute is added to the Radius Check Data attribute.

- *Rpl Add*
Select an attribute and type a value for it in the text window. If you click the Rpl Add button, the value is added to the entry definition, and the name of the attribute is added to the Radius Reply Data optional attribute which matches the grpReplyInfo attribute in the directory schema for RADIUS.
- *Chk Del*
Select the value of the attribute you want to delete from the entry definition. If you click the Chk Del button, the value is removed from the entry definition, and the name of the attribute is removed from the Radius Check Data optional attribute which matches the grpCheckInfo in the radius.mapping file.
- *Rpl Del*
Select the value of the attribute you want to delete from the entry definition. If you click the Rpl Del button, the value is removed from the entry definition, and the name of the attribute is removed from the Radius Reply Data optional attribute which matches the grpReplyInfo attribute in the directory schema for RADIUS.

The grpCheckInfo attribute contains a list of attributes that must be checked by the RADIUS server against the information supplied by the remote user. If the grpCheckInfo attribute is not present, then access is denied.

The grpReplyInfo attribute contains a list of attributes returned by the RADIUS server with an access-accept or access-reject response. It can contain connection parameters such as a PPP or SLIP profile. If the grpReplyInfo attribute is not present, the remote user can connect from any host or IP address, and through any connection protocol.

Deleting an Entry

The delete panel of Deja is used to delete entries from the directory. Figure 3.6 shows the Deja Delete panel.



Figure 3.6 Deja Delete Panel

You must have write permission for the entry you want to delete. See “Logging In” on page 49 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 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 49 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 49 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 RADIUS 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 you want to modify. See “Logging In” on page 49 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.
 - To modify attributes of the entry, see “Selecting Attributes” on page 58.
 - To change the name of the entry, use the Rename function. See “Renaming an Entry” on page 64.

The RADIUS modify attributes window features four special buttons: Chk Add, Rpl Add, Chk Del, and Rpl Del. See “Name” on page 52 for details.

3. When you have finished the modifications, click Done.

Reset

To cancel a modify operation at any time, click Reset. The entry definition is cleared from the Modify panel.

Renaming an Entry

Use Rename to modify the Relative Distinguished Name (RDN) of an entry. Figure 3.7 shows the Deja Rename panel.



Figure 3.7 Deja Rename Panel

You must have write permission for the entry you want to rename. See “Logging In” on page 49 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 a RADIUS entry in the directory. This function provides search facilities for up to three criteria. Figure 3.8 shows the Deja Search panel for RADIUS users.

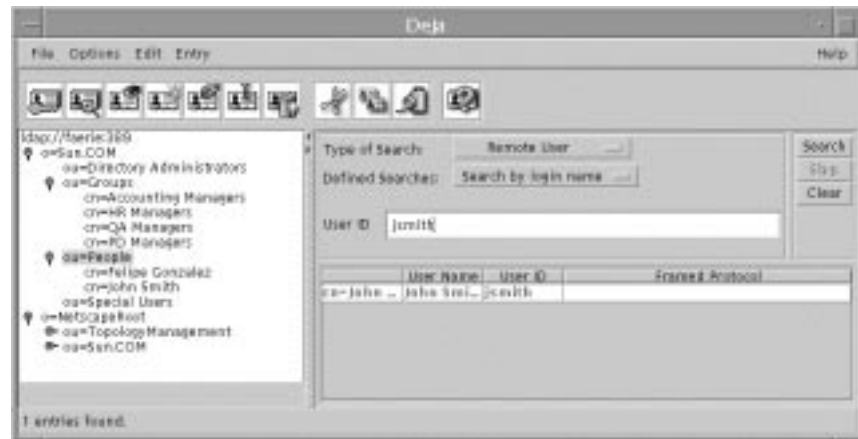


Figure 3.8 Deja Search Panel for RADIUS Users

To search for a RADIUS entry, click on the Search icon, or select Search from the Entry menu. The Search panel is displayed.

Note The types of searches available, and the categories of search results are defined in the `Deja.properties` file on the directory server. See “RADIUS Search Panel Definitions” on page 81 for information on defining searches.

The default search types are:

- Remote User search; see “Remote User Searches” on page 67
- Remote Access Server (RAS) search; see “Remote Access Server Search” on page 72
- Complex Searches; see “Complex Searches” on page 73

You can combine Remote User searches with Remote Access Server searches using AND or OR operators. You cannot combine both operators in the same search. Up to three search criteria can be defined.

Remote User Searches

There are seven searches pre-defined for remote user entries:

- **Search by login name**
Where Deja searches for entries that have the `remoteuser` objectclass and whose user id matches the text field. See “Login Name Search” on page 68.
- **Search by user name**
Where Deja searches for entries that have the `remoteuser` objectclass and whose user name matches the text field. See “User Name Search” on page 68.
- **List blocked accounts**
Where Deja searches for entries that have the `remoteuser` object class and that have one or more failed authorization accesses. See “Blocked Accounts Search” on page 69.
- **List PPP users**
Where Deja searches for entries that have the `remoteuser` object class, a PPP profile, and a PPP password set. See “List PPP Users Search” on page 69.
- **List SLIP users**
Where Deja searches for entries that have the `remoteuser` object class, a SLIP profile, and a SLIP password set. See “List SLIP Users Search” on page 70.
- **List LOGIN users**
Where Deja searches for entries that have the `remoteuser` object class, a LOGIN profile, and a LOGIN password set. See “List LOGIN Users Search” on page 71.
- **Search by name and mail**
Where Deja searches for entries that have the `remoteuser` object class and whose name and email address match the text fields. See “User Name and Mail Search” on page 71.

To define a new type of search, see “RADIUS Search Panel Definitions” on page 81.

The search root for remote user searches is stored in the `radius.mapping` file, in the `BaseDN` variable. It is also stored in the `Deja.properties` file.

Login Name Search

To perform a login name search:

1. Select Remote User from the Type of Search option button.
2. Select Search by Login Name from the Defined Searches option button.
3. Type the User ID of the entry you want to find in the search 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.

You can refine your search by combining it with RAS object class searches or other remote user object class searches. See “Complex Searches” on page 73.

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.

User Name Search

To perform a user name search:

1. Select Remote User from the Type of Search option button.
2. Select Search by User Name from the Defined Searches option button.
3. Type the user name of the entry you want to find in the search 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.

You can refine your search by combining it with RAS object class searches or other remote user object class searches. See “Complex Searches” on page 73.

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.

Blocked Accounts Search

To perform a blocked accounts search:

1. Select Remote User from the Type of Search option button.
2. Select List Blocked Accounts from the Defined Searches option button.
There are no user input fields for this search. Deja searches for entries with the following parameters:

```
objectclass = remoteuser
radiusAuthFailedAccess > RADIUS_MAX_FAIL
```

Where `RADIUS_MAX_FAIL` is defined in the `Deja.properties` file on the directory server. The default value for `RADIUS_MAX_FAIL` is 4. See “RADIUS Properties” on page 119 for information.

3. 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.
You can refine your search by combining it with RAS object class searches or other remote user object class searches. See “Complex Searches” on page 73.
4. To stop the search at any time, click the Stop button.
The search is stopped and no results are returned.
5. Click the Clear button to clear the search text field.

List PPP Users Search

To perform a PPP users search:

1. Select Remote User from the Type of Search option button.
2. Select List PPP Users from the Defined Searches option button.

There are no user input fields for this search. Deja searches for entries with the following parameters:

```
objectclass = remoteuser  
radiusPppProfile = *  
radiusPppPasswd = *
```

3. 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.

You can refine your search by combining it with RAS object class searches or other remote user object class searches. See “Complex Searches” on page 73.

4. To stop the search at any time, click the Stop button.

The search is stopped and no results are returned.

5. Click the Clear button to clear the search text field.

List SLIP Users Search

To perform a SLIP users search:

1. Select Remote User from the Type of Search option button.
2. Select List SLIP Users from the Defined Searches option button.

There are no user input fields for this search. Deja searches for entries with the following parameters:

```
objectclass = remoteuser  
radiusSlipProfile = *  
radiusSlipPasswd = *
```

3. 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.

You can refine your search by combining it with RAS object class searches or other remote user object class searches. See “Complex Searches” on page 73.

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

List LOGIN Users Search

To perform a LOGIN users search:

1. Select Remote User from the Type of Search option button.
2. Select List LOGIN Users from the Defined Searches option button.
There are no user input fields for this search. Deja searches for entries with the following parameters:

```
objectclass = remoteuser  
radiusLoginProfile = *  
radiusLoginPasswd = *
```

3. 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.
You can refine your search by combining it with RAS object class searches or other remote user object class searches. See “Complex Searches” on page 73.
4. To stop the search at any time, click the Stop button.
The search is stopped and no results are returned.
5. Click the Clear button to clear the search text field.

User Name and Mail Search

To perform a user name and mail search:

1. Select Remote User from the Type of Search option button.
2. Select Search by Name / Mail from the Defined Searches option button.
3. Type the username and email address of the entry you want to find in the search text fields.

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.

You can refine your search by combining it with RAS object class searches or other remote user object class searches. See “Complex Searches” on page 73.

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.

Remote Access Server Search

There are two searches pre-defined for RAS (NAS) entries:

- Search by RAS Name
Where Deja searches for all entries that have the objectclass RAS and whose RAS name matches the text field. See “RAS Name Search” on page 72.
- Search by RAS IP Address
Where Deja searches for all entries that have the objectclass RAS and whose RAS network address matches the text field. See “RAS IP Address Search” on page 73.

To define a new type of search, see “RADIUS Search Panel Definitions” on page 81.

The default search root for RAS searches is stored in the `radius.mapping` file, in the `BaseDN` variable. It is also stored in the `Deja.properties` file.

RAS Name Search

To perform a RAS name search:

1. Select Remote Access Server from the Type of Search option button.
2. Select Search by RAS Name from the Defined Searches option button.
3. Type the name you want to find in the search 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.

You can refine your search by combining it with remote user object class searches or other RAS searches. See “Complex Searches” on page 73.

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.

RAS IP Address Search

To perform a RAS IP address search:

1. Select Remote Access Server from the Type of Search option button.
2. Select Search by RAS IP Address from the Defined Searches option button.
3. Type the IP address you want to find in the search text field.
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.

You can refine your search by combining it with remote user object class searches or other RAS searches. See “Complex Searches” on page 73.

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.

Complex Searches

You can combine three types of search with the complex searches option:

- Remote User searches
- Remote Access Server searches

- Specify Filter; see “Search Filters” on page 74 for a description of acceptable filters

Searches can be combined with AND or OR operators. You cannot combine both operators in the same search. Up to three search criteria can be defined.

To perform a complex search:

1. Select Complex Searches from the Type of Search option button.
2. Select the first search criterion from the Remote User option button and type the search string or filter definition in the text field.
3. Click on the And or Or buttons to select the logical operator.
4. Select the second search criterion from the Remote User option button and type the search string or filter definition in the text field.
5. If you want to add a third search criterion, click the And or Or button again.
6. To remove a search criterion, click the Back button.
7. Type the Distinguished Name (DN) of the root of the tree you want to search, or select the root you want to search in the browser window and click Get from Browser.
8. 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.

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

Search Filters

Using a search filter is a way of specifying a set of entries, based on the presence of a particular attribute or attribute value. You can combine AND or OR logical operators in the same search. Use & (ampersand) for AND and | (the pipe symbol) for OR. Table 3.2 gives some examples of filters.

Table 3.2 Search Filter Examples

Filter	Definition
<code>l=London</code>	locality is "London"
<code>cn=*Rob*</code>	common name contains "Rob"
<code>(&(cn=Ch*)(cn=*Thomas*))</code>	common name starts with "Ch" and contains "Thomas"
<code>((sn=*bert*)(sn=*bort*))</code>	surname contains "bert" or "bort"
<code>(&(cn=Rob*)((cn=*Green*)(cn=*Jones*)))</code>	common name starts with "Rob" and contains "Green" or "Jones"

Search Results List

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

The headings of the search results table depend on the search. The types of searches, and the headings for search results are defined in the `Deja.properties` file on the directory server. All the headings can be modified except those for complex searches. See “RADIUS Search Panel Definitions” on page 81 for information on defining searches.

Table 3.3 shows the attributes returned for the default searches.

Table 3.3 RADIUS Search Results Lists

Search Type	Attributes
remoteUser login name	cn, uid, framedProtocol
remoteUser user name	cn, uid
remoteUser blocked accounts	cn, uid, radiusFailedAccess
remoteUser List PPP users	cn, uid
remoteUser List SLIP users	cn, uid
remoteUser List LOGIN users	cn, uid
remoteUser name/mail address	cn, uid
RAS name	cn, ipHostNumber
RAS IP address	cn, ipHostNumber
Complex searches	cn, ipHostNumber, uid, radiusFailedAccess

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. You must be authenticated as `superuser` or `root` to modify the `Deja.properties` file.

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

- “General Properties” on page 77
- “Standard LDAP Properties” on page 79
- NIS Properties
- “RADIUS Properties” on page 80

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 the meaning of the NIS 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 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 add 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.

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.

`REFERRALS_MANAGE_DSA`

With this option set to **true**, entries with the referral object class are treated like normal entries, that is the entry itself is returned in the search results. With this option set to **false**, Deja returns a search reference result. The default value is **true**.

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 **10,000** 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 **2,000** 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
#
# 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=CRAM-MD5
STANDARD_SECURITY_AUTHENTICATION=simple
```

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

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.

RADIUS Properties

You can use the RADIUS properties section of the `Deja.properties` file to define new templates for:

- Performing RADIUS searches
- Creating RADIUS entries
- Defining RADIUS profiles

RADIUS Search Panel Definitions

To add a RADIUS search to Deja, define it in the Radius Search Panel section of the `Deja.properties` file. Remote User searches are declared in the `RADIUS_RU_SEARCH` definition, and Remote Access Server searches are defined in the `RADIUS_RAS_SEARCH` definition. Each search is then defined on a separate line.

A search definition consists of:

- The search name (for example, `s_user`)
- The label that appears in the Search Type option button (for example, `RADIUS_RU_SEARCH_USER_LABEL`)
- The search definition

For example:

```
(& (objectclass=remoteuser)(uid={$uid;RADIUS_RU_UID_ATTR_LABEL$}))
```

`RADIUS_COMPLEX_SEARCH_LIST`

Contains a list of the attributes and header labels for the complex search results table. By default the `cn`, `iphostnumber` and `uid` attributes are listed.

Adding a RADIUS Remote Access Server Search

To add a RADIUS Remote Access Server search for the mail attribute:

1. Declare the search definition in the `RADIUS_RAS_SEARCH` line:

```
RADIUS_RAS_SEARCH=s_name;RADIUS_RAS_SEARCH_NAME_LABEL,  
s_addr;RADIUS_RAS_SEARCH_IPADDR_LABEL, s_mail;Search by Email
```

The name for the new search is `s_mail`, and the label that appears in the Search Type option button is Search by Email.

2. Define the search:

```
RADIUS_RAS_FILTER.s_mail=(& (objectclass=nas)  
(uid={$uid;Email;string$}))
```

The expression `{ $uid;Email;string$ }` tells Deja that for this search, the user input is a text string (`string`), the label to appear by the text field is Email (`Email`), and that the search text string is a user id (`uid`).

3. Define the headings for the search results table:

```
RADIUS_RAS_LIST.s_mail= cn;RADIUS_RAS_CN_ATTR_LABEL, uid;Email
```

If you do not specify a `RADIUS_RAS_LIST` for the search, the default headings are used (`RADIUS_RAS_LIST.default`).

4. Close Deja and restart it.

Your search type is added to the RADIUS Remote Access Server Search panel.

The following code example is an extract of the `Deja.properties` file showing the RADIUS search definitions:

```
# Radius SEARCH PANEL

# Searches defined for Remote Users

RADIUS_RU_SEARCH=s_user;RADIUS_RU_SEARCH_USER_LABEL,
s_name;RADIUS_RU_SEARCH_NAME_LABEL,
l_bl_acc;RADIUS_RU_LIST_BLOCKED_ACCOUNTS_LABEL ,
l_ppp;RADIUS_RU_LIST_PPP_USER_LABEL,
l_slip;RADIUS_RU_LIST_SLIP_USER_LABEL,
l_login;RADIUS_RU_LIST_LOGIN_USER_LABEL,
s_n_u;RADIUS_RU_SEARCH_NAME_UID_LABEL

# Associated filters for Remote User searches

RADIUS_RU_FILTER.s_user= (&
(objectclass=remoteuser)(uid={$uid;RADIUS_RU_UID_ATTR_LABEL$}))

RADIUS_RU_FILTER.s_name= (&
(objectclass=remoteuser)(cn={$cn;RADIUS_RU_CN_ATTR_LABEL$}))

RADIUS_RU_FILTER.l_bl_acc= (&
(objectclass=remoteuser)(radiusAuthFailedAccess>=$RADIUS_MAX_FAIL))

RADIUS_RU_FILTER.l_ppp= (&
(objectclass=remoteuser)(radiusPppProfile=*)(radiusPppPasswd=*))

RADIUS_RU_FILTER.l_slip= (&
(objectclass=remoteuser)(radiusSlipProfile=*)(radiusSlipPasswd=*))

RADIUS_RU_FILTER.l_login= (&
(objectclass=remoteuser)(radiusLoginProfile=*)(radiusLoginPasswd=*))

RADIUS_RU_FILTER.s_n_u= (&
(objectclass=remoteuser)(cn={$cn;RADIUS_RU_CN_ATTR_LABEL$})(uid={$uid;R
ADIUS_RU_UID_ATTR_LABEL$}))
```

```

# Attributes to be included (listed) in the search results

RADIUS_RU_LIST.s_user=      cn;RADIUS_RU_CN_ATTR_LABEL,
uid;RADIUS_RU_UID_ATTR_LABEL,
framedProtocol;RADIUS_RU_FRAMEDPROTOCOL_ATTR_LABEL

RADIUS_RU_LIST.l_bl_acc=    cn;RADIUS_RU_CN_ATTR_LABEL,
uid;RADIUS_RU_UID_ATTR_LABEL,
radiusAuthFailedAccess;RADIUS_RU_RADIUSAUTHFAILEDACCESS_ATTR_LABEL

RADIUS_RU_LIST.default=     cn;RADIUS_RU_CN_ATTR_LABEL,
uid;RADIUS_RU_UID_ATTR_LABEL


# Searches defined for RAS (Remote Access Servers)

RADIUS_RAS_SEARCH=s_name;RADIUS_RAS_SEARCH_NAME_LABEL,
s_addr;RADIUS_RAS_SEARCH_IPADDR_LABEL


# Associated filters for NAS searches

RADIUS_RAS_FILTER.s_name=   (&
(objectclass=NAS)(cn={$cn;RADIUS_RAS_CN_ATTR_LABEL$}))

RADIUS_RAS_FILTER.s_addr=   (&
(objectclass=NAS)(iphostnumber={$iphostnumber;RADIUS_RAS_IPHOSTNUMBER_A
TTR_LABEL;ipaddr$}))


# Attributes to be included (listed) in the search results

RADIUS_RAS_LIST.default=    cn;RADIUS_RAS_CN_ATTR_LABEL,
iphostnumber;RADIUS_RAS_IPHOSTNUMBER_ATTR_LABEL


# Attributes to be listed in case of a complex search

RADIUS_COMPLEX_SEARCH_LIST=cn;RADIUS_CN_ATTR_LABEL,
iphostnumber;RADIUS_RAS_IPHOSTNUMBER_ATTR_LABEL,
uid;RADIUS_RU_UID_ATTR_LABEL

```

RADIUS Create Panel Definitions

You can define alternate names for attributes that are displayed in the Choose Attributes list of the RADIUS Create panel. You can also restrict user input to one of the four basic input types (**int**, **string**, **crypt** and **ipaddr**). The default *input_type* is **string**.

RADIUS_RU_ADD_COMMON defines attributes for Remote User Entries that are common to all remote user profiles, and RADIUS_RAS_ADD_COMMON defines attributes for Remote Access Server entries that are common to all remote user profiles. The syntax of an attribute definition is:

```
RADIUS_RAS_ADD_COMMON= attribute_name;label;input_type, ...
```

Where:

attribute_name is the name of an attribute

label is the name you want to appear in the Choose Attributes list instead of the attribute name

input_type is one of the four basic input types (**int**, **string**, **crypt** and **ipaddr**). The default *input_type* is **string**.

The following code example is an extract of the `Deja.properties` file showing the RADIUS create panel definitions:

```
# Radius ADD PANEL

RADIUS_RU_ADD_COMMON=      uid;RADIUS_RU_UID_ATTR_LABEL,
grpCheckInfo;RADIUS_RU_GRP_CHECKINFO_ATTR_LABEL,
grpReplyInfo;RADIUS_RU_GRP_REPLYINFO_ATTR_LABEL,
framedIPAddress;RADIUS_RU_FRAMED_IPADDRESS_LABEL;ipaddr,
userPassword;RADIUS_RU_USER_PASSWORD_LABEL;crypt

RADIUS_RAS_ADD_COMMON=
iphostNumber;RADIUS_RAS_IP_HOSTNUMBER_ATTR_LABEL;ipaddr,
sharedKey;RADIUS_RAS_SHARED_KEY_LABEL;crypt
```

RADIUS Profiles

Three RADIUS Remote User profiles are supplied in the default `Deja.properties` file. There are no Remote Access Server profiles 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 / RADIUS_RAS_PROFILE
```

Specifies the RADIUS profiles available in Deja. The default profiles are SLIP, PPP and LOGIN. The syntax is:

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

Where:

profile_name is the name of the profile

label is the label that appears in the Create or Modify panels.

`RADIUS_RU_ADD.profile_name` / `RADIUS_RAS_ADD.profile_name`

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

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

`RADIUS_RAS_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

input_type is one of the four basic input types **int**, **string**, **crypt** and **ipaddr**). The default input_type is **string**.

The following code example is an extract of the `Deja.properties` file showing the RADIUS Profile Definitions:

```
# Profiles defined for Remote Users (RU)

RADIUS_RU_PROFILE=      ppp_p;RADIUS_RU_PPP_PROFILE_LABEL,
                        slip_p;RADIUS_RU_SLIP_PROFILE_LABEL,
                        login_p;RADIUS_RU_LOGIN_PROFILE_LABEL

# Mandatory RU profile attributes (you can edit the next line by ADDING
# attributes, but
# NEVER erase the attributes that are given by default)

RADIUS_RU_ADD.ppp_p=
radiusppppprofile;RADIUS_RU_RADIUSPPPPPROFILE_ATTR_LABEL;int,
radiusPppPasswd;RADIUS_RU_RADIUSPPPPASSWD_ATTR_LABEL;crypt

RADIUS_RU_ADD.slip_p=
radiusSlippprofile;RADIUS_RU_RADIUSSLIPPROFILE_ATTR_LABEL;int,
radiusSlipPasswd;RADIUS_RU_RADIUSSLIPPASSWD_ATTR_LABEL;crypt

RADIUS_RU_ADD.login_p=
radiusLoginprofile;RADIUS_RU_RADIUSLOGINPROFILE_ATTR_LABEL;int,
radiusLoginPasswd;RADIUS_RU_RADIUSLOGINPASSWD_ATTR_LABEL;crypt

# Profiles defined for Remote Access Servers (RAS)

#RADIUS_RAS_PROFILE=      no defined profiles
```

Setting Deja Properties

```
# Mandatory RAS profile attributes
#RADIUS_RAS_ADD.??=      no defined profiles
```

RADIUS/LDAP Information Mapping

This chapter describes how RADIUS information is stored in the LDAP directory. It provides a list of RADIUS object classes and attributes, and shows the mapping between a RADIUS dictionary and directory objects.

This chapter contains the following sections:

- “Attribute Mapping” on page 87
- “RADIUS Schema” on page 91

Attribute Mapping

The RADIUS attributes and values defined in all of the dictionary files provided with the Solaris Extensions for Netscape Directory Server 4.11 are mapped onto LDAP attributes. This mapping is defined in the `radius.mapping` file located in `/etc/opt/SUNWconn/ldap/current/mapping`.

There is a one-to-one correlation between RADIUS attributes and LDAP attributes. Therefore, the mapping syntax is very simple. You can easily add proprietary RADIUS attributes to the default mapping.

A copy of the default `radius.mapping` file is stored in the directory `/opt/SUNWconn/ldap/default`. You should keep this file as a reference copy, and not modify it.

Default Mapping

Table 4.1 shows the one-to-one correspondence between RADIUS attributes and LDAP attributes. The table also indicates the origin of each RADIUS attribute. There are several kinds of RADIUS attributes:

- Standard attributes
- Vendor-specific attributes
- Attributes defined by the Solaris Extensions for Netscape Directory Server 4.11 product

Standard RADIUS attributes are specified in RFC 2138 *Remote Authentication Dial In User Service (RADIUS)* and RFC 2139 *RADIUS Accounting*. Vendor-specific attributes are defined by NAS vendors and supplied in the dictionary file they provide with their equipment. There aren't any vendor-specific attributes in the dictionary file provided with Solaris Extensions for Netscape Directory Server 4.11.

The LDAP attributes that allow the Netscape Directory Server to store RADIUS information in the directory database are defined in the `radius.at.conf` file. This file is stored in the `/opt/SUNWconn/ldap/default/schema` directory.

The LDAP object classes that allow the Netscape Directory Server to store RADIUS information in the directory database are defined in the `radius.oc.conf` file. This file is also stored in the `/opt/SUNWconn/ldap/default/schema` directory.

The purpose of all RADIUS attributes and object classes is described in "RADIUS Schema" on page 91.

Table 4.1 RADIUS-to-LDAP Attribute Mapping

RADIUS Attribute	Origin	LDAP Attribute
User-Name	RFC 2138	uid
Crypt-Password	Solaris Extensions	userPassword
CHAP-Password	RFC 2138	chapPassword
NAS-IP-Address	RFC 2138	ipHostNumber
NAS-Identifier	RFC 2138	authNASIdentifier
NAS-Port	RFC 2138	authHostPortNumber
Service-Type	RFC 2138	authServiceProtocol
Framed-Protocol	RFC 2138	framedProtocol
Framed-IP-Address	RFC 2138	framedIPAddress

RADIUS Attribute	Origin	LDAP Attribute
Framed-IP-Netmask	RFC 2138	ipNetmaskNumber
Framed-Routing	RFC 2138	framedRouting
Filter-Id	RFC 2138	authFilterId
Framed-MTU	RFC 2138	framedMTU
Framed-Compression	RFC 2138	framedCompression
Login-IP-Host	RFC 2138	ipLoginHost
Login-Service	RFC 2138	authLoginService
Login-TCP-Port	RFC 2138	ipLoginPort
Reply-Message	RFC 2138	authReplyMessage
Callback-Number	RFC 2138	userCallbackNumber
Callback-Id	RFC 2138	userCallbackId
Framed-Route	RFC 2138	framedRoute
Framed-IPX-Network	RFC 2138	ipxNetworkNumber
State	RFC 2138	authState
Session-Timeout	RFC 2138	sessionTimeoutNumber
Idle-Timeout	RFC 2138	idleTimeoutNumber
Termination-Action	RFC 2138	authTerminationAction
Called-Station-Id	RFC 2138	authCalledStationId
Calling-Station-Id	RFC 2138	authCallingStationId
NAS-Port-Type	RFC 2138	authHostPortType
Port-Limit	RFC 2138	authPortLimit
Acct-Status-Type	RFC 2139	acctStatusType
Acct-Delay-Time	RFC 2139	acctDelayTime
Acct-Input-Octets	RFC 2139	acctInputOctet
Acct-Input-Packets	RFC 2139	acctInputPacket
Acct-Output-Octets	RFC 2139	acctOutputOctet
Acct-Output-Packets	RFC 2139	acctOutputPacket
Acct-Session-Id	RFC 2139	acctSessionId
Acct-Authentic	RFC 2139	acctAuthentic
Acct-Session-Time	RFC 2139	acctSessionTime
Acct-Terminate-Cause	RFC 2139	acctTerminateCause
Expiration	Solaris Extensions	expirationDate

Attribute Mapping

RADIUS Attribute	Origin	LDAP Attribute
Auth-Type	Solaris Extensions	authType
Menu	Solaris Extensions	authStartMenuId
Termination-Menu	Solaris Extensions	authStopMenuId
Prefix	Solaris Extensions	authPrefixName
Suffix	Solaris Extensions	authSuffixName
user-check	Solaris Extensions	grpCheckInfo
user-reply	Solaris Extensions	grpReplyInfo
Login-Profile	Solaris Extensions	radiusLoginProfile
PPP-Profile	Solaris Extensions	radiusPppProfile
SLIP-Profile	Solaris Extensions	radiusSlipProfile
Login-Passwd	Solaris Extensions	radiusLoginPasswd
PPP-Passwd	Solaris Extensions	radiusPppPasswd
SLIP-Passwd	Solaris Extensions	radiusSlipPasswd
Login-Expiration	Solaris Extensions	radiusLoginExpiration
PPP-Expiration	Solaris Extensions	radiusPppExpiration
SLIP-Expiration	Solaris Extensions	radiusSlipExpiration
Auth-Failed-Access	Solaris Extensions	radiusAuthFailedAccess
Dynamic-Session-Counter	Solaris Extensions	dynamicSessionCounter
Dynamic-SessionId	Solaris Extensions	dynamicSessionId
Dynamic-IPAddress	Solaris Extensions	dynamicIPAddress
Dynamic-IPAddr-Binding	Solaris Extensions	DynamicIPAddrBinding
Dictionary-File	Solaris Extensions	dictionaryFile
AcctAttr-File	Solaris Extensions	acctattrFile
PAM-Service-Name	Solaris Extensions	pamServiceName

Extending the Default Mapping

You can change the default mapping provided in the `radius.mapping` file to suit your own needs. To extend the default mapping to add RADIUS attributes that are not included in the default mapping, you must create a RADIUS/LDAP mapping definition.

To create a RADIUS/LDAP mapping definition:

1. Create an LDAP attribute for each missing RADIUS attribute.

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

2. Add the RADIUS-LDAP attribute pair to the `/etc/opt/SUNWconn/ldap/current/radius.mapping` file, using a text editor.

Make sure you add it in both the Import section and the Export section of the file. You need to be logged in as `root` to perform this operation.

3. Restart the `ns-slapd` daemon so that the modifications to the schema are taken into account.

4. Restart the `dsradiusd` daemon so that the new mapping file is taken into account. As `root`, type the following commands:

```
# /opt/SUNWconn/ldap/sbin/dsradius stop
# /opt/SUNWconn/ldap/sbin/dsradius start
```

5. Run the `dejasync` utility. As `root` type:

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

For details on the options of the `dejasync` utility, refer to “`dejasync`” on page 121. You must run `dejasync` if you want to use the Deja tool to modify RADIUS entries in the directory.

Note You can modify an existing RADIUS object class to add a new attribute. However, the best approach is to create a sub-class of an existing object class to hold the new attribute.

RADIUS Schema

This section describes the RADIUS schema. It lists the object classes and attributes that are required to use the RADIUS service. The RADIUS schema is automatically added to the directory server schema when you run the RADIUS initialization script, `setup_rad`, as described in “Initializing RADIUS” on page 21.

RADIUS Object Classes

RADIUS object classes are defined in the `radius.oc.conf` file. This file is located in the `/opt/SUNWconn/ldap/default/schema` directory.

The RADIUS service includes the following specific object classes:

- `nas`: used to create NAS entries in the directory database
- `remoteUser`: used to create remote user entries in the directory database
- `radiusServer`: used to describe hosts that are RADIUS servers

nas

Description: Defines a Network Access Server used in the context of RADIUS authentication.

Superior object class: `device`

Mandatory attributes: `iphostNumber`, `sharedKey`

Optional attributes: `acctattrFile`, `dictionaryFile`

remoteUser

Description: In the context of RADIUS authentication, used to define remote users who access the network through a Network Access Server (NAS). The `remoteUser` object class is an auxiliary object class. This means that it can be used with any structural object class, for example the `person` or `organizational person` object class. The `uid` attribute is mandatory because it is always passed in the connection request transmitted by the NAS to the RADIUS server. It is the key attribute used in the search filter applied by the RADIUS server to look for the remote user's entry in the directory. The optional attributes are the LDAP translation of the RADIUS attributes. They define all the possible connection parameters that can be passed in a connection request transmitted by the NAS to the RADIUS server.

Superior object class: `top`

Mandatory attribute: `uid` (`userid`)

Optional attributes: `acctAuthentic`, `acctDelayTime`, `acctInputOctet`, `acctInputPacket`, `acctOutputOctet`, `acctOutputPacket`, `acctSessionId`, `acctSessionTime`, `acctStatusType`, `acctTerminateCause`, `authCalledStationId`, `authCallingStationId`, `authFilterId`,

authHostPortNumber, authHostPortType, authLoginService, authNASIdentifier, authPortLimit, authPrefixName, authReplyMessage, authServiceProtocol, authType, authStartMenuId, authState, authStopMenuId, authSuffixName, authTerminationAction, chapPassword, cn (commonName), dynamicSessionCounter, dynamicSessionId, dynamicIPAddress, dynamicIPAddrBinding, expirationDate, framedCompression, framedIPAddress, framedMTU, framedRoute, framedRouting, framedProtocol, grpCheckInfo, grpReplyInfo, idleTimeoutNumber, ipHostNumber, ipLoginHost, ipLoginPort, ipNetmaskNumber, ipxNetworkNumber, pamServiceName, radiusLoginProfile, radiusPppProfile, radiusSlipProfile, radiusAuthFailedAccess, radiusLoginExpiration, radiusLoginPasswd, radiusPppExpiration, radiusPppPasswd, radiusSlipExpiration, radiusSlipPasswd, sessionTimeoutNumber, userCallbackId, userCallbackNumber, userPassword.

radiusServer

Description: This object class is reserved for future use.

Superior object class: applicationProcess

Mandatory attributes: host, sharedKey

Optional attributes: dictionaryFile, acctattrFile, authHostPortNumber, acctHostPortNumber, radiusServerRealm, radiusServerFlags

RADIUS Attributes

RADIUS attributes are defined in the `radius.at.conf` file. This file is located in the `/opt/SUNWconn/ldap/default/schema` directory.

All attributes defined in the RADIUS 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)
- UTC time (utctime)

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

acctattrFile

Description: Specifies the name of the dynamic accounting attributes file to be used to interpret the dynamic accounting information received from the NAS described by the entry.

Syntax: ces

Contained in object class: nas, radiusServer

acctAuthentic

Description: Used in RADIUS accounting requests to indicate how the user described by the entry was authenticated.

Syntax: ces

Contained in object class: remoteUser

acctDelayTime

Description: Used in RADIUS accounting requests to indicate for how long the NAS has been trying to send an accounting report. The delay is deducted from the time of arrival of the report to determine the actual time at which the event occurred.

Syntax: ces

Contained in object class: remoteUser

acctInputOctet

Description: Used in RADIUS accounting requests to indicate the number of octets received during the provision of service.

Syntax: ces

Contained in object class: remoteUser

acctInputPacket

Description: Used in RADIUS accounting requests to indicate the number of packets received during the provision of service.

Syntax: ces

Contained in object class: remoteUser

acctOutputOctet

Description: Used in RADIUS accounting requests to indicate the number of octets sent during the provision of service.

Syntax: ces

Contained in object class: remoteUser

acctOutputPacket

Description: Used in RADIUS accounting requests to indicate the number of packets sent during the provision of service.

Syntax: ces

Contained in object class: remoteUser

acctSessionId

Description: Used in RADIUS accounting to provide a unique accounting ID. It is used to match start and stop records for the same session.

Syntax: ces

Contained in object class: remoteUser

acctSessionTime

Description: Used in RADIUS accounting to indicate the number of seconds during which the user described by the entry has received service.

Syntax: ces

Contained in object class: remoteUser

acctStatusType

Description: Used in RADIUS accounting to indicate whether the current report marks the beginning of service (start) or the end (stop).

Syntax: ces

Contained in object class: remoteUser

acctTerminateCause

Description: Used in RADIUS accounting to indicate how a session was terminated.

Syntax: ces

Contained in object class: remoteUser

authCalledStationId

Description: Indicates the phone number called by the user to request access through a NAS.

Syntax: ces

Contained in object class: remoteUser

authCallingStationId

Description: Indicates the phone number from which the user called to request access through a NAS.

Syntax: ces

Contained in object class: remoteUser

authFilterId

Description: Indicates the name of the filter list for the user described by the entry.

Syntax: ces

Contained in object class: remoteUser

authHostPortNumber

Description: Indicates the physical port number of the NAS that is authenticating the user.

Syntax: ces

Contained in object classes: remoteUser, radiusServer

authHostPortType

Description: Indicates the type of physical port number of the NAS that is authenticating the user.

Syntax: ces

Contained in object class: remoteUser

authLoginService

Description: Indicates the service that should be used to connect the user to the login host.

Syntax: ces

Contained in object class: remoteUser

authNASIdentifier

Description: Contains a string that identifies the NAS that transmitted an access request.

Syntax: ces

Contained in object class: remoteUser

authPortLimit

Description: Sets the maximum number of ports to be provided by the NAS to the user.

Syntax: ces

Contained in object class: remoteUser

authPrefixName

Description: Used internally by the RADIUS server to distinguish between the user name to be processed for authentication and a possible prefix. In some cases, the connection protocol can add a prefix to the user's name, for example, ppp%jsmith.

Syntax: ces

Contained in object class: remoteUser

authReplyMessage

Description: Contains text that the NAS can display to the user.

Syntax: cis

Contained in object class: remoteUser

authServiceProtocol

Description: Indicates the type of service requested by the user.

Syntax: ces

Contained in object class: remoteUser

authStartMenuId

Description: This attribute is used internally by the RADIUS server.

Syntax: ces

Contained in object class: remoteUser

authState

Description: A state attribute sent by the RADIUS server to the NAS. The NAS must send it back unchanged in the reply to the server. This attribute is single-valued.

Syntax: ces

Contained in object class: remoteUser

authStopMenuId

Description: Used internally by the RADIUS server.

Syntax: ces

Contained in object class: remoteUser

authType

Description: Indicates to the RADIUS server how passwords are stored, so that the password supplied by the user can be compared correctly against the password stored under the user's entry in the directory. Possible values for this attribute are:

- **Crypt-Local** - Specifies that passwords are stored encrypted
- **Local** - Specifies that passwords are stored in clear text
- **System** - Specifies that passwords are maintained in `/etc/passwd`

Syntax: ces

Contained in object class: remoteUser

authSuffixName

Description: Used internally by the RADIUS server to distinguish between the user name to process for authentication and a possible suffix. In some cases, the domain name can be added to the user's name, for example, `jsmith@eng.xyz.com`.

Syntax: ces

Contained in object class: remoteUser

authTerminationAction

Description: Indicates the action to perform by the NAS when the service session is finished.

Syntax: ces

Contained in object class: remoteUser

chapPassword

Description: Contains the response value provided by a PPP Challenge Handshake Authentication Protocol (CHAP) user in response to a challenge. This attribute is single-valued.

Syntax: ces

Contained in object class: remoteUser

dictionaryFile

Description: Specifies the dictionary to be used by the RADIUS server when it receives a request from the NAS described by the entry.

Syntax: ces

Contained in object class: nas, radiusServer

dynamicIPAddressBinding

Description: When RADIUS accounting is activated, associates the dynamicIPAddress and the dynamicSessionId assigned to the remote user.

Syntax: cis

Contained in object class: remoteUser

dynamicIPAddress

Description: When RADIUS accounting is activated, the IP address assigned to the remote user is recorded in the user's entry using this attribute. This attribute is created when the session begins, and removed when the session ends.

Syntax: cis

Contained in object class: remoteUser

dynamicSessionCounter

Description: When RADIUS accounting is activated, the number of concurrent open sessions for a remote user is recorded in the user's entry using this attribute. This attribute is removed when the user ends the last session. This attribute is single-valued.

Syntax: int

Contained in object class: remoteUser

dynamicSessionId

Description: When RADIUS accounting is activated, the session identifier assigned to the remote user for a particular session is recorded in the user's entry using this attribute. This identifier is used in to open and close the accounting report for the session.

Syntax: cis

Contained in object class: remoteUser

expirationDate

Description: Indicates the expiration date for the password stored in the userPassword attribute. The expirationDate attribute is single-valued.

Syntax: ces

Contained in object class: remoteUser

framedCompression

Description: Indicates a compression protocol to be used for the link.

Syntax: ces

Contained in object class: remoteUser

framedIPAddress

Description: Indicates the address to be configured for the user.

Syntax: ces

Contained in object class: remoteUser

framedMTU

Description: Indicates the maximum transmission unit (MTU) to be configured for the user, when it is not negotiated by some other means (such as PPP).

Syntax: ces

Contained in object class: remoteUser

framedProtocol

Description: Indicates the framing to be used for framed access.

Syntax: ces

Contained in object class: remoteUser

framedRoute

Description: Provides routing information to be configured for the user on the NAS. Not to be confused with the framedRouting attribute.

Syntax: ces

Contained in object class: remoteUser

framedRouting

Description: Indicates the routing method for the user, when the user is a router to a network. Not to be confused with the framedRoute attribute.

Syntax: ces

Contained in object class: remoteUser

grpCheckInfo

Description: Contains a list of attributes (except uid) that must be checked by the RADIUS server against the information supplied by the remote user. If this attribute is not present, then access is denied. This attribute is used internally by the server.

Syntax: ces

Contained in object class: remoteUser

grpReplyInfo

Description: Contains a list of attributes returned by the RADIUS server with an access-accept or access-reject response. It can contain connection parameters such as a PPP or SLIP profile. If this attribute is not present, the remote user can connect from any host or IP address, and through any connection protocol. This attribute is used internally by the server.

Syntax: ces

Contained in object class: remoteUser

idleTimeoutNumber

Description: Sets the maximum number of consecutive seconds that the connection can remain idle before the session is terminated.

Syntax: ces

Contained in object class: remoteUser

ipLoginHost

Description: Indicates the system with which to connect the user, when the `authLoginService` attribute is included in the connection request.

Syntax: `cis`

Contained in object class: `remoteUser`

ipLoginPort

Description: Indicates the TCP port with which the user is to be connected, when the `authLoginService` attribute is included in the connection request.

Syntax: `cis`

Contained in object class: `remoteUser`

ipxNetworkNumber

Description: Indicates the IPX network number to be configured for the user.

Syntax: `cis`

Contained in object class: `remoteUser`

pamServiceName

Description: Specifies the name of the service that provides the PAM module. If you want to use PAM authentication with the RADIUS server, set the value of this attribute to be **radius**. You must also add **pamServiceName** to the list of attributes in the `grpCheckInfo` attribute. This attribute is single-valued.

Syntax: `ces`

Contained in object class: `remoteUser`

radiusAuthFailedAccess

Description: Created dynamically in a remote user's entry when an access request is rejected. This counter is incremented by 1 at each failed attempt. The user account is blocked when this counter reaches the blocking value specified in the configuration (by default, 4). This attribute is single-valued.

Syntax: ces

Contained in object class: remoteUser

radiusLoginExpiration

Description: Indicates the expiration date for the password stored in the radiusLoginPasswd attribute. This attribute is single-valued.

Syntax: ces

Contained in object class: remoteUser

radiusLoginPasswd

Description: Password provided by the remote user to gain access to the network through the LOGIN protocol. This attribute is single-valued.

Syntax: ces

Contained in object class: remoteUser

radiusLoginProfile

Description: Flag with value 0 or 1. Value 1 enables checking of the password supplied by the user against the password stored in the radiusLoginPasswd attribute. Value 0 disables this check. This attribute is single-valued.

Syntax: ces

Contained in object class: remoteUser

radiusPppExpiration

Description: Indicates the expiration date for the password stored in the radiusPppPasswd attribute. This attribute is single-valued.

Syntax: ces

Contained in object class: remoteUser

radiusPppPasswd

Description: Password provided by the remote user to gain access to the network through the PPP protocol. This attribute is single-valued.

Syntax: ces

Contained in object class: remoteUser

radiusPppProfile

Description: Flag with value 0 or 1. Value 1 enables checking of the password supplied by the user against the password stored in the radiusPppPasswd attribute. Value 0 disables this check. This attribute is single-valued.

Syntax: ces

Contained in object class: remoteUser

radiusServerFlags

Description: Reserved for future use.

Syntax: ces

Contained in object class: radiusServer

radiusServerRealm

Description: Reserved for future use.

Syntax: ces

Contained in object class: radiusServer

radiusSlipExpiration

Description: Indicates the expiration date for the password stored in the radiusSlipPasswd attribute. This attribute is single-valued.

Syntax: ces

Contained in object class: remoteUser

radiusSlipPasswd

Description: Password provided by the remote user to gain access to the network through the SLIP protocol. This attribute is single-valued.

Syntax: ces

Contained in object class: remoteUser

radiusSlipProfile

Description: Flag with value 0 or 1. Value 1 enables checking of the password supplied by the user against the password stored in the radiusSlipPasswd attribute. Value 0 disables this check. This attribute is single-valued.

Syntax: ces

Contained in object class: remoteUser

sessionTimeoutNumber

Description: Sets the maximum number of seconds of service to be provided to the user described in the entry before the session is shut down.

Syntax: ces

Contained in object class: remoteUser

sharedKey

Description: Specifies the shared secret used by the network access server (NAS) described by the entry during RADIUS authentication. This attribute is single-valued.

Syntax: ces

Contained in object classes: nas, radiusServer

userCallbackId

Description: Indicates a name of a place to be called. This attribute is interpreted by the NAS.

Syntax: ces

Contained in object class: remoteUser

userCallbackNumber

Description: Indicates a dialing string to use for callback to provide service to the user.

Syntax: ces

Contained in object class: remoteUser

userid

Description: The uid, or userid (mandatory), is always passed in the connection request transmitted by the NAS to the RADIUS server. It is the key attribute used in the search filter applied by the RADIUS server to look for the remote user's entry in the directory.

Syntax: cis

Contained in object class: remoteUser

userPassword

Description: The password that the user described by the entry uses to gain access to the entry. This password is automatically encrypted by the directory server.

Contained in object class: `remoteUser`

RADIUS Schema

Command & File Reference

This chapter describes the daemons, commands and files that provide RADIUS service.

acctattr

Synopsis

The location of the `acctattr` file is:

```
/etc/opt/SUNWconn/ldap/current/acctattr
```

Description

The `acctattr` file contains a list of RADIUS attributes that are recorded for each remote connection when dynamic accounting is enabled. The attributes listed in this file are recorded in the directory entry of the remote user.

The RADIUS attributes listed in the `acctattr` file must belong to the dictionary file supported by the NAS.

Other accounting parameters that are automatically recorded when dynamic accounting is enabled are:

- `dynamicIPAddress`
- `dynamicSessionId`
- `dynamicSessionCounter`
- `dynamicIPAddressBinding`

If you add attributes to the `acctattr` file, you must ensure that there is a corresponding LDAP attribute for each RADIUS attribute. The mapping of the RADIUS attribute to the LDAP attribute must be declared in the `radius.mapping` file.

See Also

See “dictionary” on page 123, “radius.mapping” on page 134

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 114
- “Standard LDAP Properties” on page 115
- “NIS Properties” on page 117
- “RADIUS Properties” on page 119

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 the `ipHostNumber` attribute type to be in 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 **10,000** 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 **2,000** 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=airius_remote_users,c=us`.

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=airius_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 “`dejasync`” on page 121, “`radius.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`.

It creates or updates tokens in the `Deja.properties` file and also backs up the `Deja.properties` file.

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

`nis.mapping` File

The `dejasync` command gets the list of NIS maps managed by `Deja` from the `Deja.properties` file. 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`

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 “radius.mapping” on page 134, and “Deja.properties” on page 112

dictionary

Synopsis

The location of the `dictionary` file is:

`/etc/opt/SUNWconn/ldap/current/dictionary`

Description

The dictionary file is used for communication between the RADIUS server and the NAS. A RADIUS dictionary file contains RADIUS attribute and value pairs. A number of these attributes are defined in RFC 2138 *Remote Authentication Dial In User Service (RADIUS)*, and RFC 2139 *RADIUS Accounting*. However, NAS vendors have also defined proprietary attributes.

The dictionary file contains standard RADIUS attribute definitions and non-protocol attributes that are used internally by the RADIUS server. Internal attributes are listed in Table 2.1 on page 35.

Dictionary files are often provided by NAS vendors such as Ascend, Cisco, Shiva or Bay Networks. Solaris Extensions for Netscape Directory Server 4.11 can be also be used with these vendor-supplied dictionaries.

dsnmpcfg

Synopsis

The syntax of the `dsnmpcfg` command is:

```
/opt/SUNWconn/ldap/sbin/dsnmpcfg { install | configure | remove }
```

Description

The `dsnmpcfg` script is used by the `pkgadd` utility to initialize the `dsnmprad` daemon provided with Solaris Extensions for Netscape Directory Server 4.11. It is also used by the `pkgrm` utility to stop the daemon. The `dsnmprad` daemon is an SNMP agent that collects statistics on RADIUS traffic.

The `dsnmpcfg` script is also used by to declare the hosts to which the SNMP agent must send SNMP traps. It does this by modifying the `trap-recipients` variable in the `dsnmprad.conf` file.

Options

install

Called by `pkgadd` when the `SUNWsds` package is installed to generate the initial SNMP configuration. It creates the configuration file `dsnmprad.conf` if it does not already exist. It then starts the RADIUS SNMP daemon, `dsnmprad`.

configure

Used by the administrator to configure the SNMP configuration file, `dsnmprad.conf`. The administrator is prompted to provide the names of the hosts to which the SNMP daemon must send SNMP events (traps). When the configuration is complete, the `dsnmprad` script restarts the SNMP daemon `dsnmprad`.

remove

Called by `pkgrm` when the `SUNWsds` package is removed to remove the SNMP configuration. It stops `dsnmprad`, then removes the `dsnmprad.conf` file and the log files.

See Also

See “`dsnmprad`” on page 125

dsnmprad

Synopsis

The syntax of the `dsnmprad` daemon is:

```
/opt/SUNWconn/ldap/lib/dsnmprad [-h] [-a target-agent-host] [-b
boots-file] [-c config-file] [-i poll-interval] [-l log-file] [-p port]
[-s log-size] [-t timeout] [-T trace-level] [-y internal-auth-port] [-z
internal-acc-port]
```

Description

The dsnmprad daemon is an SNMP agent that implements RFC 2619 *RADIUS Authentication Server MIB* and RFC 2621 *RADIUS Accounting Server MIB*.

The dsnmprad daemon implements the SNMP GET and GET-NEXT commands. It does not implement the SNMP SET command.

The dsnmprad configuration files, `dsnmprad.conf` and `dsnmprad.boots` are located by default in the `/etc/opt/SUNWconn/ldap/current` directory. If you move them to a new location, you must specify it on the command line (options `-b` and `-c`) when you start dsnmprad.

The dsnmprad daemon gets the list of manager hosts (recipients of SNMP traps) from the `dsnmprad.conf` file.

Options

`-h`

Online help (usage).

`-a target-agent-host`

Default is localhost.

`-b boots-file`

The default boots file is

`/etc/opt/SUNWconn/ldap/current/dsnmprad.boots`.

`-c config-file`

Specifies the configuration file containing the list of hosts that are recipients of SNMP traps. The default file is

`/etc/opt/SUNWconn/ldap/current/dsnmprad.conf`.

`-i poll-interval`

Default poll interval is 10 seconds.

`-l log-file`

Default log file is `/tmp/radiusd.log`.

-p *port*

Indicates the UDP port number on which the SNMP agent listens for traps. The default port number is 161.

-s *log-size*

Indicates the maximum size of the log file. The default value is 100,000 bytes.

-t *timeout*

Default timeout is 4000000 microseconds.

-T *trace-level*

The trace level parameter is in the range 0 to 4. The default value is 0.

-y *internal-auth-port*

Default is 34654.

-z *internal-acc-port*

Default is 34754.

See Also

See “dsnmppcfg” on page 124

dsradius

Synopsis

The syntax of the dsradius script is:

```
/opt/SUNWconn/ldap/sbin/dsradius { start | stop | refresh }
```

Description

The `dsradius` script can be used to start and stop the `dsradiusd` daemon. It is also used to force the `dsradiusd` daemon to re-read the configuration without stopping and restarting. You must perform a restart or a refresh of the `dsradius` daemon whenever you change the RADIUS configuration, or when you add, delete or modify NAS entries in the directory.

Options

start | stop | refresh

Use the appropriate keyword to start or stop the `dsradiusd` daemon. The refresh keyword forces `dsradiusd` to re-read the `dsradiusd.conf` configuration file, or to the NAS branch of the directory tree.

See Also

See “`dsradiusd`” on page 128, “`dsradiusd.conf`” on page 132

dsradiusd

Synopsis

The syntax of the `dsradiusd` daemon is:

```
/opt/SUNWconn/ldap/lib/dsradiusd [-a acct_dir] [-c conf_file] [-d
config_dir] [-f failed_password_blocking_number] [-h] [-i ip_address]
[-I] [-l log_file] [-m mapping_file] [-o] [-p authentication_udp_port]
[-P accounting_udp_port] [-q max_outstanding_requests] [-s] [-t
max_seconds_in_queue] [-v] [-x trace-level] [-y min_threads] [-z
max_threads]
```


Description

The `dsradiusd` daemon is the RADIUS server daemon. It is an authentication server for remote users connecting to a network through a Network Access Server (NAS), also called Remote Access Server (RAS), that uses Netscape Directory Server to store information about remote users and NAS'. The NAS communicates with `dsradiusd` to check the information provided in the connection request against the information stored in the directory.

RADIUS (Remote Access Dialup User Service) is the protocol used by the `dsradiusd` daemon to authenticate remote users who connect to the network.

The `dsradiusd` daemon also provides accounting information on remote user connections. When the daemon is started, it forks two processes, one to handle authentication, to other to handle accounting.

The `dsradiusd` daemon can be started with the `dsradius` script. The startup parameters are saved in the `radius.mapping` file.

Options

`-a acct_dir`

Specifies the directory where accounting information is stored. The default is `/var/opt/SUNWconn/ldap/radacct`. This directory contains one subdirectory per NAS, named after the NAS hostname, or the NAS IP host number.

`-c conf_file`

Specifies the name of the configuration file. The default is `dsradiusd.conf`.

`-d config_dir`

Specifies the directory where the configuration file is stored. The default is `/etc/opt/SUNWconn/ldap/current`.

`-f failed_password_blocking_limit`

Specifies the number of times a wrong password can be supplied before the user account is blocked. By default, the user account is blocked after four unsuccessful attempts.

- h
Displays the usage message.
- i *ip_address*
Specifies the IP address of the RADIUS server. Use this option if the IP address of the host machine and of the RADIUS server are different, for example if the RADIUS server runs on a machine that has multiple network interfaces.
- I
Automatically adds dynamic connection attributes to the entry of the user when an accounting-start signal is received. These attributes are removed when an accounting-stop signal is received. The dynamic connection attributes are the assigned IP address, accounting session ID, session counter (number of simultaneous sessions opened by the same remote user), and all attributes listed in `acctattr`. For this option to work, RADIUS accounting must be activated. This means that the Network Access Server (NAS) must be configured to send accounting packets to the RADIUS server.
- l *log_file*
Specifies the file where logging information is saved. The default is `/var/opt/SUNWconn/ldap/log/radius.log`.
- m *mapping_file*
Specifies the `radius.mapping` file used by the `dsradiusd` daemon.
- O
Specifies that the server will accept all zeros in the authenticator that is passed between the NAS and the RADIUS server. This allows compatibility with NAS devices that support older versions of the RADIUS protocol.
- p *authentication_udp_port*
Specifies the UDP port on which the authentication process of the `dsradiusd` daemon is started. The default port number for the authentication process is **1645**. Due to later standardization, the standard port number is **1812**.
- P *accounting_udp_port*
Specifies the UDP port on which the accounting process of the `dsradiusd` daemon is started. The default port number for the accounting process is **1646**. Due to later standardization, the standard port number is **1813**.

-q *max_outstanding_requests*

Indicates the maximum number of connection requests that can be queued. The default is **5000**.

-s

Starts the dsradiusd daemon in single process mode for debugging.

-t *max_seconds_in_queue*

Sets the timeout on a connection request in seconds. The default value is **30**.

-v

Starts the dsradiusd daemon in verbose mode. The output is displayed on screen.

-x *trace-level*

Specifies a trace level for the dsradiusd daemon. When the daemon is running with this option, it does not fork or dissociate from the terminal. The trace level can be the sum of:

- 1=trace (the default)
- 2=args
- 4=config
- 8=mapping
- 64=memory

-y *min_threads*

Specifies the minimum number of threads used by the dsradiusd daemon to process connection requests. The default value is **3**.

-z *max_threads*

Specifies the maximum number of threads used by the dsradiusd daemon to process connection requests. The default value is **1024**.

See Also

See “dsradiusd.conf” on page 132

dsradiusd.conf

Synopsis

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

```
/etc/opt/SUNWconn/ldap/default/dsradiusd.conf
```

Description

The `dsradiusd.conf` file contains configuration parameters for the RADIUS service. The configuration parameters held in this file are different from those defined in the mapping file, `radius.mapping`.

The `dsradiusd.conf` file contains the following configuration parameters:

- `LDAP_Server`
Specifies the hostname of the LDAP server. The default value is **localhost**.
- `LDAP_Port`
Specifies the TCP port number that the LDAP server runs on. The default value is **389**.
- `LDAP_Bind_dn`
Specifies the DN used to bind to the LDAP directory for performing searches and modifying the LDAP database. The example value provided as a placeholder in `dsradiusd.conf` is **cn=radiusadmin**.
- `LDAP_Password`
Specifies the password used to bind to the LDAP directory for performing searches and modifying the LDAP database. The example value provided as a placeholder in `dsradiusd.conf` is **secret**.
- `LogLevel`
Specifies the log level. You can select one of the following log levels:
 - 0 = none
 - 1 = trace

- 2 = trace and translation
- 3 = trace, translation, and other debug information

The default is **0**.

- `LogSize`
Specifies the number of kilobytes before the log file automatically changes over to the next log file. The default value is **1000**.
- `LogDir`
Specifies the log directory where `dsrcadius.log` is located. The default value is `/var/opt/SUNWconn/ldap/log`.

See Also

See “radius.mapping” on page 134, “dsradiusd” on page 128

radius.at.conf

Synopsis

The location of the `radius.at.conf` file is:

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

Description

The `radius.at.conf` file contains schema information used for storing RADIUS authentication and accounting information in the Netscape Directory Server. It contains a list of LDAP attributes required to use the Netscape Directory Server to store RADIUS information.

Attribute definitions in the `radius.oc.conf` file contain:

- The name of the attribute

radius.mapping

- The OID of the attribute
- The attribute syntax
- If the attribute is single valued, the keyword *single*

For example, the definition of the dictionaryFile attribute is:

```
attribute dictionaryFile 1.3.6.1.4.1.42.2.27.1.1.74 ces single
```

For information on the possible attribute syntaxes, refer to “RADIUS Attributes” on page 93.

See Also

See “radius.oc.conf” on page 136.

radius.mapping

Synopsis

The location of the `radius.mapping` file is:

```
/opt/SUNWconn/ldap/default/mapping/radius.mapping
```

Description

The `radius.mapping` file contains:

- Configuration parameters for the RADIUS service
- RADIUS-to-LDAP mapping information

Configuration Parameters

The configuration information stored in the `radius.mapping` file is at the beginning of the file, under the section entitled *Common*.

The following configuration variables are defined:

- `Max_allowed_failures`
This parameter determines blocking mode. It defines the number of permitted consecutive failures to authenticate a user based on the password provided. Any further attempt is systematically blocked, even if the connection parameters supplied are correct. The count is reset on first success. The default value for this parameter is **4**. To disable blocking mode, set this parameter to **0**.
When an account is blocked, you must manually reset the `authFailedAccess` attribute in the remote user's entry to **0**.
- `Dynamic`
This parameter determines whether dynamic accounting data is recorded in the LDAP directory (see "Configuring Dynamic Accounting" on page 36). It can have one of two values **on** or **off**. By default, dynamic accounting is **off**.
- `Authentication_Port`
The authentication port number for RADIUS processes. The default port number is **1645**. Due to later standardization, the standard port number is **1812**. This port number is defined in the `radius.mapping` file but is commented out.
- `Accounting_Port`
The accounting port number for RADIUS processes. The default port number is **1646**. Due to later standardization, the standard port number is **1813**. This port number is defined in the `radius.mapping` file but is commented out.
- `Accounting_dir`
The directory where accounting information is stored. The default directory is `/var/opt/SUNWconn/ldap/radacct`. Note that there is a typographical error in this variable in the `radius.mapping` file. Do not correct it because the RADIUS server would be unable to read it.
- `Max_wait_b4_reject`
This parameter defines a client timeout on a RADIUS request. It determines the maximum time a client will wait for a response from the RADIUS server. By default it is **58** seconds.
- `Time_limit`
This parameter defines a RADIUS server timeout on LDAP search operations. It must be *strictly less than* the value of the `Max_wait_b4_reject` parameter. By default it is **50** seconds.

radius.oc.conf

Note Blocking mode and dynamic accounting settings are not taken into account when the RADIUS search is performed on a referral server.

Mapping Information

The `radius.mapping` file defines the mapping of attributes in the RADIUS dictionary file to LDAP attributes. There is a one-to-one mapping between RADIUS attributes and LDAP attributes. This mapping is shown in detail in Table 4.1 on page 88.

This mapping information is used by the RADIUS server when it performs LDAP searches in the directory to check authentication information provided in remote user connection requests. It is also used by the RADIUS server to write accounting information into the remote user entries in the directory.

The syntax and semantics of the mapping information in the `radius.mapping` file is described in Appendix A, “Mapping Syntax and Semantics.”

See Also

See “`dsradiusd.conf`” on page 132.

radius.oc.conf

Synopsis

The location of the `radius.oc.conf` file is:

```
/opt/SUNWconn/ldap/default/schema/radius.oc.conf
```


Description

The `radius.oc.conf` file contains schema information used for storing RADIUS authentication and accounting information in the Netscape Directory Server. It contains a list of LDAP object classes required to use the Netscape Directory Server to store RADIUS information.

Object class definitions in the `radius.oc.conf` file contain:

- The name of the superior object class, introduced by the keyword *superior*
- A list of mandatory attributes, introduced by the keyword *requires*
- A list of optional attributes, introduced by the keyword *allows*

For example, the definition of the `radiusServer` object class is:

```
objectclass radiusServer
    superior applicationProcess
    requires
        host,
        sharedKey
    allows
        dictionaryFile,
        acctattrFile,
        authHostPortNumber,
        acctHostPortNumber
```

See Also

See “`radius.at.conf`” on page 133.

setup_rad

setup_rad

Synopsis

The syntax of the `setup_rad` script is:

```
/opt/SUNWconn/ldap/sbin/setup_rad -d
```

Description

The `setup_rad` script initializes the RADIUS server. It registers the RADIUS console with the Netscape Console and prompts you for all the information required by the directory server to allow connections from the RADIUS server.

Options

`-d`

Undoes the setup. This option removes the RADIUS console icon from the Netscape Console, and disables the RADIUS configuration.



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 RADIUS schema, refer to “RADIUS Schema” on page 91.

This chapter contains the following sections:

- “File Structure” on page 140
- “Mapping Semantics” on page 140
- “Mapping Syntax” on page 146

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 146.

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 148.

Table A.1 Summary of Mapping File Keywords

Mapping Semantics

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 143.
	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_OBJECTCLASS” 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 145.
	LINE	Mandatory	Defines decomposition of input information. See “LINE” on page 144.
	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 144.
	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 144.

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 naming context 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 147.

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 147.
- **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
```

In this example, 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.

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.

Mapping Syntax

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

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
```

Mapping Syntax

Index

A

- access
 - temporary 28, 30, 31
 - through specified NAS 30, 31
- access control rules
 - Deja 49
- access denial 24, 135
- accounting
 - dynamic 36
- accounting log
 - RADIUS 16
- Accounting_Port parameter 25, 135
- acctSessionId attribute 95
- acctattr file 37
 - default 39
 - location 111
 - reference 111
- acctattrFile attribute 94
- acctAuthentic attribute 94
- acctDelayTime attribute 94
- acctInputOctet attribute 94
- acctInputPacket attribute 95
- acctOutputOctet attribute 95
- acctOutputPacket attribute 95
- acctSessionTime attribute 96
- acctStatusType attribute 96
- acctTerminateCause attribute 96
- Accounting_dir parameter 25, 135
- ALL_FILTER keyword 144
- attribute
 - acctattrFile 94
 - acctAuthentic 94
 - acctDelayTime 94
 - acctInputOctet 94
 - acctInputPacket 95
 - acctOutputOctet 95
 - acctOutputPacket 95
 - acctSessionId 95
 - acctSessionTime 96
 - acctStatusType 96
 - acctTerminateCause 96
 - assigning a value 58
 - authCalledStationId 96
 - authCallingStationID 96
 - authFilterId 97
 - authHostPortNumber 97
 - authHostPortType 97
 - authLoginService 97
 - authNASIdentifier 97
 - authPortLimit 98
 - authPrefixName 98
 - authReplyMessage 98
 - authServiceProtocol 98
 - authStartMenuId 98
 - authState 99
 - authStopMenuId 99
 - authSuffixName 99
 - authTerminationAction 100
 - authType 99
 - chapPassword 100
 - deleting a value 59
 - dictionaryFile 100
 - dynamicIPaddress 101
 - dynamicIPaddressBinding 100
 - dynamicSessionCounter 101
 - dynamicSessionId 101
 - expirationDate 101
 - framedCompression 102
 - framedIPaddress 102
 - framedMTU 102
 - framedProtocol 102
 - framedRoute 102

- framedRouting 103
- grpCheckInfo 103
- grpReplyInfo 103
- idleTimeoutNumber 103
- ipLoginHost 104
- ipLoginPort 104
- ipxNetworkNumber 104
- modifying a value 59
- pamServiceName 104
- radiusAuthFailedAccess 105
- radiusLoginExpiration 105
- radiusLoginPasswd 105
- radiusLoginProfile 105
- radiusPppExpiration 106
- radiusPppPasswd 106
- radiusPppProfile 106
- radiusServerFlags 106
- radiusServerRealm 106
- radiusSlipExpiration 107
- radiusSlipPasswd 107
- radiusSlipProfile 107
- sessionTimeoutNumber 107
- sharedKey 108
- userCallbackId 108
- userCallbackNumber 108
- userid 108
- userPassword 109
- attribute syntax 93
- attributes
 - dynamic accounting 36
 - selecting 58
- authCalledStationId attribute 96
- authCallingStationId attribute 96
- authentication
 - PAM module 19
- Authentication_Port parameter 25, 135
- authFilterId attribute 97
- authHostPortNumber attribute 97
- authHostPortType attribute 97
- authLoginService attribute 97
- authNASIdentifier attribute 97
- authPortLimit attribute 98

- authPrefixName attribute 98
- authReplyMessage attribute 98
- authServiceProtocol attribute 98
- authStartMenuId attribute 98
- authState attribute 99
- authStopMenuId attribute 99
- authSuffixName attribute 99
- authTerminationAction attribute 100
- authType attribute 99

B

- BASE_DN keyword 143
- blocked accounts search 69
- blocking mode 24, 135

C

- chapPassword attribute 100
- command
 - dejasync 121, 122
- commands
 - dsnmpcfg 124
 - ypcat 144
 - ypmatch 144
- complex searches 73
- configuration
 - dynamic accounting 36
 - RADIUS server 24, 26
- configuration files
 - dsradiusd.conf 24
 - radius.mapping 24
- configuration parameter
 - Accounting_Port 25, 135
 - Accounting_dir 25, 135
 - Authentication_Port 25, 135
 - Dynamic 25, 135
 - Max_allowed_failures 24, 135
 - Max_wait_b4_reject 25, 135
 - RADIUS 24
 - Time_limit 25, 135

- configuration parameters
 - RADIUS 26
- connection properties 52
- conventions, in this book 14
- copy operation 62
- copying an entry
 - from the View window 56
- create parameters
 - Deja.properties file 83
- creating an entry 56
 - selecting attributes 58
- crypt syntax 118
- cut operation 62

D

- daemons
 - dsnmprad 125
 - dsradiusd 128
- DC_NAMING keyword 145
- Deja 46, 53, 54
 - access control rules 49
 - connecting to another server 54
 - connection properties 52
 - copying an entry 62
 - creating an entry 56
 - cutting an entry 62
 - deleting an entry 60
 - display options 51
 - general parameters 77
 - host name 53
 - logging in 49
 - modifying an entry 64
 - new window, opening 53
 - pasting an entry 63
 - port number 53
 - properties 51
 - reconnecting to server 53
 - renaming an entry 64
 - reset operation 64
 - restoring an entry 63
 - search results 75
 - searching for an entry 66

- selecting attributes 58
- starting 48
- toolbar icons 46
- user properties 52
- View window 55
- viewing an entry 55

- Deja.properties file
 - adding a RADIUS search 81
 - create parameters 83
 - general parameters 77
 - location 112
 - RADIUS profiles 84
 - reference 112
- dejasync command 122
 - reference 121
 - syntax 121
- delete operation 60
- dictionary file 34
 - default 35
 - internal server attributes 35
 - location 123
 - reference 123
 - specifying 35
- dictionaryFile attribute 100
- dsnmprad command
 - reference 124
 - syntax 124
- dsnmprad daemon 39
 - reference 125
 - syntax 125
- dsradius script
 - reference 127
 - syntax 127
- dsradiusd daemon 21
 - reference 128
 - syntax 128
- dsradiusd.conf file 24
 - location 132
 - reference 132
- dynamic accounting 25, 135
 - attribute
 - creating 38

- attributes 36
 - examples 37
- enabling 36
- parameters 37
- RADIUS 36

Dynamic parameter 25, 135

dynamicIPAddress attribute 101

dynamicIPAddressBinding attribute 100

dynamicSessionCounter attribute 101

dynamicSessionId attribute 101

E

entry 55

- copying 62
- cutting 62
- deleting 60
- highlighting 56
- naming 57
- pasting 63

entry modifying

- assigning attribute values 58
- deleting attribute values 59
- modifying attribute values 59

entry renaming 64

environment variable

- JAVA_HOME 49

exclude function 151

expirationDate attribute 101

F

files

- acctattr 111
- Deja.properties 112
- dictionary 123
- dsradiusd.conf 132
- radius.at.conf 133
- radius.mapping 24, 134
- radius.oc.conf 136

fonts, in this book 14

framedCompression attribute 102

framedIPAddress attribute 102

framedMTU attribute 102

framedProtocol attribute 102

framedRoute attribute 102

framedRouting attribute 103

G

getrdn function 151

- mapping 151

grpCheckInfo attribute 103

grpReplyInfo attribute 103

H

host name 53

I

icons 46

idleTimeoutNumber attribute 103

initialization

- RADIUS server 21

instances2string function 150

int syntax 118

internal attributes 35

ipaddr syntax 118

ipLoginHost attribute 104

ipLoginPort attribute 104

ipxNetworkNumber attribute 104

J

JAVA_HOME environment variable 49

K

keywords

- mapping 141

L

- LINE keyword 144
- list LOGIN users search 71
- list PPP users search 69
- list SLIP users search 70
- log
 - RADIUS accounting 16
- logging in
 - Deja 49
- login name search 68

M

- MAP_NAME 143
- MAP_NAME keyword 143
- Mapping
 - RADIUS/LDAP 88
- mapping 143
 - ALL_FILTER 144
 - BASE_DN 143
 - DC_NAMING 145
 - exclude function 151
 - extending 90
 - instances2string function 150
 - keyword summary 141
 - LINE 144, 145
 - MATCH_FILTER 144
 - PRIVATE_OBJECTCLASSES 143
 - RADIUS/LDAP 88
 - split function 149
 - string2instances function 149
 - trim function 150
- mapping file
 - Common section 143
 - Condense section 148
 - Dynamic section 144, 147
 - Export section 145
 - Extract section 147
 - Import section 145
 - radius.mapping 87
 - semantics 140
 - structure 140

- syntax 146

- MATCH_FILTER keyword 144
- Max_allowed_failures parameter 24, 135
- Max_wait_b4_reject parameter 25, 135
- modify operation 64
 - cancel 64

N

- naming
 - RADIUS entry 57
- nis.mapping file
 - synchronization 122

O

- object class
 - radiusServer 93

P

- PAM module 19
- pamServiceName attribute 104
- parameters
 - dynamic accounting 37
 - RADIUS configuration 24, 26
- paste operation 63
- port number 53
- PRIVATE_OBJECTCLASSES keyword 143
- properties
 - Deja 51

R

- RADIUS
 - accounting log 16
 - attributes 34
 - blocking mode 24, 135
 - dictionary file 34
 - dynamic accounting 36
 - attributes 36
- RADIUS access

- temporary 28, 30, 31
- through specified NAS 30, 31
- virtual domain 32
- RADIUS accounting 16
 - monitoring 41
- RADIUS authentication
 - monitoring 40
- RADIUS configuration
 - files 24
 - parameters 24, 26
- RADIUS entry
 - search 66
 - search results 75
 - searching 66
- RADIUS functions
 - naming an entry 57
- RADIUS profiles
 - Deja.properties file 84
- RADIUS schema 91
- RADIUS searches
 - processing order 33
- RADIUS server 13
 - configuration 24, 26
 - initialization 21
 - internal attributes 35
 - statistics 39
- RADIUS statistics
 - displaying 42
- radius.at.conf file
 - location 133
 - reference 133
- radius.mapping file 24, 87
 - location 134
 - reference 134
 - synchronization 122
 - virtual domains 32
- radius.oc.conf file
 - location 136
 - reference 136
- RADIUS/LDAP mapping 88
 - extending 90

- radiusAuthFailedAccess attribute 105
- radiusLoginExpiration attribute 105
- radiusLoginPasswd attribute 105
- radiusLoginProfile attribute 105
- radiusPppExpiration attribute 106
- radiusPppPasswd attribute 106
- radiusPppProfile attribute 106
- radiusServer object class 93
- radiusServerFlags attribute 106
- radiusServerRealm attribute 106
- radiusSlipExpiration attribute 107
- radiusSlipPasswd attribute 107
- radiusSlipProfile attribute 107
- RAS entry
 - searching 72
- RAS IP address search 73
- RAS name search 72
- reconnecting Deja 53
- refresh
 - Deja 54
- refreshing display 54
- remote user
 - searching 67
- rename operation 64
- reset operation 64
- restore operation 63

S

- schema 91
- scripts
 - dsradius 127
 - setup_rad 138
- search
 - parameters 81
- search operation 66
 - RADIUS entry 66
 - search results 75

- searching
 - blocked accounts 69
 - complex search 73
 - login name 68
 - LOGIN users 71
 - PPP users 69
 - RAS
 - IP address 73
 - name 72
 - RAS entry 72
 - RAS IP address 73
 - RAS name 72
 - remote user 67
 - blocked accounts 69
 - list LOGIN users 71
 - list PPP users 69
 - list SLIP users 70
 - login name 68
 - user name 68
 - user name and mail 71
 - SLIP users 70
 - user name 68
 - user name and mail 71
- sessionTimeoutNumber attribute 107
- setup_rad script
 - reference 138
 - syntax 138
- sharedKey attribute 108
- SNMP agent
 - dsnmprad 39
- split function 149
- statistics 39
 - accounting service 41
 - authentication service 40
- string syntax 118
- string2instances function 149
- styles, in this book 14
- synchronization
 - nis.mapping file 122
 - radius.mapping file 122
- syntax
 - crypt 118

- int 118
- ipaddr 118
- of LDAP attributes 93
- string 118

T

- temporary access 28, 30
- terms, in this book 14
- Time_limit parameter 25, 135
- timeout 25, 135
- trim function 150

U

- user name and mail search 71
- user name search 68
- user profile
 - selecting 52
- user properties 52
- userCallbackId attribute 108
- userCallbackNumber attribute 108
- userid attribute 108
- userPassword attribute 109

V

- View window
 - closing 55
 - copy entry 56
 - displaying 55
 - highlighting an entry 56
- viewing 55
- viewing an entry 55
- virtual domain 32

W

- window, closing 53

Y

ypcat command 144

ypmatch command 144