

NIS+ and FNS Administration Guide

2550 Garcia Avenue
Mountain View, CA 94043
U.S.A



SunSoft
A Sun Microsystems, Inc. Business

© 1995 Sun Microsystems, Inc. 2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A.

All rights reserved. This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Portions of this product may be derived from the UNIX[®] system, licensed from UNIX Systems Laboratories, Inc., a wholly owned subsidiary of Novell, Inc., and from the Berkeley 4.3 BSD system, licensed from the University of California. Third-party software, including font technology in this product, is protected by copyright and licensed from Sun's Suppliers.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

TRADEMARKS

Sun, Sun Microsystems, the Sun logo, SunSoft, the SunSoft logo, Solaris, SunOS, OpenWindows, DeskSet, ONC, ONC+, NFS, AdminTool, and AdminSuite are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd. OPEN LOOK is a registered trademark of Novell, Inc. PostScript and Display PostScript are trademarks of Adobe Systems, Inc.

All SPARC trademarks are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries. SPARCcenter, SPARCcluster, SPARCcompiler, SPARCdesign, SPARC811, SPARCengine, SPARCprinter, SPARCserver, SPARCstation, SPARCstorage, SPARCworks, microSPARC, microSPARC-II, and UltraSPARC are licensed exclusively to Sun Microsystems, Inc. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK[®] and Sun[™] Graphical User Interfaces were developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUI's and otherwise comply with Sun's written license agreements.

X Window System is a trademark of X Consortium, Inc.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN, THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. SUN MICROSYSTEMS, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAMS(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.



Contents

Preface.....	xxix
<i>Part 1—NIS+ Introduction and Overview</i>	
1. Introduction to Name Services.....	1
What Is a Name Service?.....	2
DNS.....	8
FNS.....	8
NIS.....	8
NIS Architecture.....	9
NIS Maps.....	10
NIS+.....	11
What NIS+ Can Do for You.....	12
How NIS+ Differs From NIS.....	13
NIS+ Security.....	16
NIS+ and the Name Service Switch.....	17
Solaris 1.x and NIS-Compatibility Mode.....	17

NIS+ Administration Commands	18
NIS+ API	20
2. The NIS+ Namespace	21
NIS+ Files and Directories	22
Structure of the NIS+ Namespace	22
Directories	24
Domains	25
Servers	27
How Servers Propagate Changes	28
NIS+ Clients and Principals	30
Principal	30
Client	31
The Cold-Start File and Directory Cache	32
An NIS+ Server Is Also a Client	36
Naming Conventions	37
NIS+ Domain Names	40
Directory Object Names	40
Tables and Group Names	40
Table Entry Names	41
Host Names	41
NIS+ Principal Names	42
Accepted Name Symbols	42
NIS+ Name Expansion	43
NIS_PATH Environment Variable	43

3. NIS+ Tables and Information	45
NIS+ Table Structure	45
Columns and Entries	47
Search Paths	48
Ways to Set Up Tables	50
How Tables Are Updated	52
4. Security Overview	53
Solaris Security—Overview	53
NIS+ Security—Overview	56
NIS+ Principals	58
NIS+ Security Levels	58
Security Levels and Password Commands	59
NIS+ Authentication and Credentials—Introduction	59
User and Machine Credentials	60
DES versus LOCAL Credentials	60
DES Credentials	60
LOCAL Credentials	61
User Types and Credential Types	62
NIS+ Authorization and Access—Introduction	63
Authorization Classes	63
The Owner Class	64
The Group Class	65
The World Class	66
The Nobody Class	66

Authorization Classes and the NIS+ Object Hierarchy	66
NIS+ Access Rights	67
The NIS+ Administrator	68
NIS+ Password, Credential, and Key Commands	69
<i>Part 2—Administering NIS+</i>	
5. Administering NIS+ Credentials	73
How Credentials Work	74
Credential versus Credential Information	74
Authentication Components	75
How Principals are Authenticated	75
Credentials Preparation Phase	76
Login Phase—Detailed Description	76
Request Phase—Detailed Description	77
The DES Credential in Detail	79
DES Credential Secure RPC Netname	80
DES Credential Verification Field	80
How the DES Credential Is Generated	80
Secure RPC Password versus Login Password Problem	82
Cached Public Keys Problems	83
Where Credential-Related Information Is Stored	84
The Cred Table in Detail	85
Creating Credential Information	86
The nisaddcred Command	87
Related Commands	88

How nisaddcred Creates Credential Information	89
LOCAL Credential Information	89
DES Credential Information	89
The Secure RPC Netname and NIS+ Principal Name	90
Creating Credential Information for the Administrator	91
Creating Credential Information for NIS+ Principals	91
For User Principals—Example	93
Using a Dummy Password and chkey—Example	93
Creating in Another Domain—Example	95
For Workstations—Example	96
Administering NIS+ Credential Information	96
Updating Your Own Credential Information	97
Removing Credential Information	97
6. Administering NIS+ Keys	99
Keylogin	99
Changing Keys for a NIS+ Principal	100
Changing the Keys	102
Changing Root Keys From Root	102
Changing Root Keys From Another Machine	104
Changing the Keys of a Root Replica from the Replica	104
Changing the Keys of a Nonroot Server	105
Updating Public Keys	105
The nisupdkeys Command	105
Updating Public Keys Arguments and Examples	106

Updating IP Addresses	107
7. Administering NIS+ Access Rights	109
Introduction to Authorization and Access Rights	110
Authorization Classes—Review	110
Access Rights—Review	110
Concatenation of Access Rights	111
How Access Rights Are Assigned and Changed	112
Specifying Different Default Rights	112
Changing Access Rights to an Existing Object	112
Table, Column, and Entry Security	113
Table, Column, Entry Example	114
Rights at Different Levels	115
Where Access Rights Are Stored	117
Viewing an NIS+ Object’s Access Rights	118
Default Access Rights	119
How a Server Grants Access Rights to Tables	119
Specifying Access Rights in Commands	120
Syntax for Access Rights	121
Class, Operator, and Rights Syntax	121
Syntax for Owner and Group	122
Syntax for Objects and Table Entries	123
Displaying NIS+ Defaults—The <code>nisdefaults</code> Command	124
Setting Default Security Values	125
Displaying the Value of <code>NIS_DEFAULTS</code>	126

Changing Defaults.....	126
Resetting the Value of NIS_DEFAULTS.....	127
Specifying Nondefault Security Values at Creation Time.....	127
Changing Object and Entry Access Rights.....	128
Using nischmod to Add Rights.....	128
Using nischmod to Remove Rights.....	129
Specifying Column Access Rights.....	130
Setting Column Rights When Creating a Table.....	130
Adding Rights to an Existing Table Column.....	131
Removing Rights to a Table Column.....	132
Changing Ownership of Objects and Entries.....	132
Changing Object Owner With nischown.....	133
Changing Table Entry Owner With nischown.....	133
Changing an Object or Entry's Group.....	134
Changing an Object's Group With nischgrp.....	134
Changing a Table Entry's Group With nischgrp.....	135
8. Administering Passwords.....	137
Using Passwords.....	138
Logging In.....	138
The Login incorrect Message.....	139
The password expired Message.....	139
The will expire Message.....	140
The Permission denied Message.....	140
Changing Your Password.....	140

Password Change Failures	142
Choosing a Password	142
Password Requirements	142
Bad Choices for Passwords	143
Good Choices for Passwords	143
Administering Passwords.....	144
nsswitch.conf File Requirements.....	144
The nispasswd Command	144
The yppasswd Command	145
The passwd Command.....	145
passwd and the nsswitch.conf File	145
The passwd Command and “NIS+ Environment” ..	147
The passwd Command and Credentials.....	147
The passwd Command and Permissions	147
The passwd Command and Keys	148
The passwd Command and Other Domains	148
The nistbladm Command.....	148
nistbladm and Shadow Column Fields	149
nistbladm And the Number of Days	152
Related Commands.....	154
Displaying Password Information.....	154
Changing Passwords.....	156
Changing Your Own Password	156
Changing Someone Else’s Password	156

Changing Root's Password	157
Locking a Password.	157
Unlocking a Password	158
Managing Password Aging	158
Forcing Users to Change Passwords	159
Setting a Password Age Limit.	160
Setting Minimum Password Life	161
Establishing a Warning Period	162
Turning Off Password Aging	163
Password Privilege Expiration	164
Specifying Maximum Number of Inactive Days	166
Setting Password Aging Criteria for Multiple Users.	168
Specifying Password Criteria and Defaults	168
The /etc/defaults/passwd File	168
Password Failure Limits	171
9. Administering NIS+ Groups	173
Related Commands	174
Specifying Group Members	174
Using <code>niscat</code> With Groups	175
Listing the Object Properties of a Group.	176
The <code>nisgrpadm</code> Command	177
Creating an NIS+ Group	178
Deleting an NIS+ Group	179
Adding Members to an NIS+ Group	179

Listing the Members of an NIS+ Group	180
Removing Members From an NIS+ Group.....	181
Testing for Membership in an NIS+ Group	182
10. Administering NIS+ Directories	183
Using the <code>niscat</code> Command With Directories.....	184
Listing the Object Properties of a Directory.....	184
The <code>nislsls</code> Command.....	184
Listing the Contents of a Directory—Terse.....	185
Listing the Contents of a Directory—Verbose	186
The <code>nismkdir</code> Command	187
Creating a Directory	187
Adding a Replica to an Existing Directory.....	189
The <code>nismrmdir</code> Command	190
Removing a Directory.....	190
Disassociating a Replica From a Directory.....	191
The <code>nismrm</code> Command.....	191
Removing Nondirectory Objects	192
The <code>rpc.nisd</code> Command.....	192
Starting a NIS-Compatible Daemon	193
Starting a DNS-Forwarding NIS-Compatible Daemon....	194
Stopping the NIS+ Daemon	194
The <code>nisinit</code> Command.....	194
Initializing a Client	195
Initializing the Root Master Server	196

The <code>nis_cachemgr</code> Command	196
Starting the Cache Manager	197
The <code>nisshowcache</code> Command	197
Displaying the Contents of the NIS+ Cache	197
The <code>nisping</code> Command	198
Displaying the Time of the Last Update	199
Pinging Replicas	199
Checkpointing a Directory	200
The <code>nislog</code> Command	201
Displaying the Contents of the Transaction Log	201
The <code>nischttl</code> Command	203
Changing the Time-to-Live of an Object	204
Changing the Time-to-Live of a Table Entry	205
11. Administering NIS+ Tables	207
The <code>nistbladm</code> Command	208
Creating a New Table	209
Deleting a Table	211
Adding an Entry to a Table	211
Using the <code>-a</code> Option	211
Using the <code>-A</code> Option	213
Modifying a Table Entry	213
Removing a Single Entry From a Table	214
Removing Multiple Entries From a Table	215
The <code>niscat</code> Command	216

Displaying the Contents of a Table	217
Displaying the Object Properties of a Table or Entry	217
The <code>nismatch</code> and <code>nisgrep</code> Commands	219
About Regular Expressions	220
Searching the First Column	221
Searching a Particular Column	222
Searching Multiple Columns	222
The <code>nisl</code> Command	222
Syntax	223
Creating a Link	223
The <code>nissetup</code> Command	224
Expanding a Directory Into an NIS+ Domain	225
Expanding a Directory Into an NIS-Compatible Domain	225
The <code>nisaddent</code> Command	226
Loading Information From a File	228
Loading Data From an NIS Map	229
Dumping the Contents of an NIS+ Table to a File	231
12. The Name Service Switch	233
About the Name Service Switch	233
Format of the <code>nsswitch.conf</code> File	234
Search Criteria	235
Switch Status Messages	236
Switch Action Options	236
Default Search Criteria	236

What if the Syntax Is Wrong?	238
Auto_home and Auto_master	238
Timezone	238
Comments in nsswitch.conf Files	238
The nsswitch.conf Template Files	239
Switch Template File Examples	240
Default nsswitch.conf File	241
DNS Forwarding	241
DNS Forwarding for NIS+ Clients	242
DNS Forwarding for NIS Clients	242
Adding Compatibility With +/- Syntax	242
13. Removing NIS+	245
Removing NIS+ From a Client Machine	245
Removing NIS+ That Was Installed Using nisclient ...	245
Removing NIS+ That Was Installed Using NIS+ Commands	246
Removing NIS+ From a Server	246
Removing NIS+ From a Server	247
Removing the NIS+ Namespace	248
 <i>Part 3—Administering FNS</i>	
14. Administering FNS in NIS+	253
Setting Up FNS	254
Estimating Resource Requirements	254
Setting Up NIS+ Service for FNS	254
Setting Up the FNS Namespace	255

Replicating FNS Service	256
Creating FNS Contexts Individually	257
Organization Context	259
All Hosts Context.....	260
Single Host Context.....	260
Host Aliases	261
All Users Context.....	261
Single User Context.....	262
Service Context	263
Printer Context.....	264
Generic Context	264
Site Context.....	265
File Context.....	266
Namespace Identifier Context	266
Managing and Examining FNS Contexts	267
Displaying the Binding.....	267
Listing the Context	269
Binding a Composite Name to a Reference	273
Removing a Composite Name	275
Renaming an Existing Binding.....	275
Destroying the Named Context	276
Managing and Examining FNS Attributes	276
Adding an Attribute	276
Deleting an Attribute.....	277

Listing an Attribute	277
Modifying an Attribute	278
Other Options	278
Maintaining Consistency Between NIS+ and FNS	278
Checking Naming Inconsistencies	279
Advanced FNS and NIS+ Issues	280
Mapping FNS Contexts to NIS+ Objects	280
Browsing FNS Structures Using NIS+ Commands	280
Checking Access Control	282
Significance of Double Slashes	283
Significance of Trailing Slash	284
Troubleshooting and Error Messages	284
15. Federating NIS+ With Global Naming Systems	285
Obtaining the NIS+ Root Reference	285
Federating NIS+ Under DNS	286
Federating NIS+ Under X.500	288
16. Administering the File System Namespace	291
The FNS File System Namespace	291
NFS File Servers	292
The Automounter	293
Creating File Contexts	294
Creating the Input File	295
Using Command-line Input	297
Advanced Input Formats	298

Multiple Locations	298
Variable Substitution	298
Backward Compatibility Input Format	299
Administering File Contexts	299
17. Administering the Printer Namespace	301
The Printer Namespace	301
Administering printer Contexts	302
Using Files	302
Using NIS	302
Using NIS+	303
 <i>Part 4—Appendices</i>	
A. Problems and Solutions	307
Namespace Administration Problems	308
Illegal Object Problems	308
nisinit Fails	309
Checkpoint Keeps Failing	309
Cannot Add User to a Group	309
Logs Grow too Large	309
Lack of Disk Space	310
Cannot Truncate Transaction Log File	310
Domain Name Confusion	310
Cannot Delete org_dir or groups_dir	311
Namespace Database Problems	311
Multiple rpc.nisd Parent Processes	311

NIS Compatibility Problems.....	312
User Cannot Log In After Password Change.....	313
nsswitch.conf File Fails to Perform Correctly.....	314
<i>Object Not Found Problems</i>	314
Syntax or Spelling Error	315
Incorrect Path.....	315
Domain Levels Not Correctly Specified	315
Object Does Not Exist	316
Lagging or Out-of-Sync Replica.....	316
Files Missing or Corrupt.....	316
Old /var/nis Filenames.....	317
Blanks in Name	317
Cannot Use Automounter.....	318
Ownership and Permission Problems	318
No Permission	319
No Credentials	319
Server Running at Security Level 0	319
User Login Same as Machine Name	319
Bad Credentials	321
Security Problems	321
“Login Incorrect” Message	321
Password Locked, Expired, or Terminated.....	322
Stale and Outdated Credential Information.....	322
Storing and Updating Credential Information.....	322

Updating Stale Cached Keys.....	324
Corrupted Credentials	327
Keysevr Failure	328
Machine Previously Was an NIS+ Client	329
No Entry in the cred Table	329
Changed Domain Name	329
When Changing a Machine to a Different Domain	329
NIS+ Password and Login Password in <code>/etc/passwd</code> File	330
Secure RPC Password and Login Passwords Are Different	330
Preexisting <code>/etc/.rootkey</code> File.....	331
Root Password Change Causes Problem	332
Slow Performance and System Hang Problems.....	332
Checkpointing	333
Variable <code>NIS_PATH</code>	333
Table Paths	333
Too Many Replicas.....	334
Recursive Groups.....	334
Large NIS+ Database Logs at Start-up	334
The Master <code>rpc.nisd</code> Daemon Died	334
No <code>nis_cachemgr</code>	335
Server Very Slow at Start-up After NIS+ Installation	335
<code>niscat</code> Returns: Server busy. Try Again	336
NIS+ Queries Hang After Changing Host Name	336
System Resource Problems.....	337

Insufficient Memory	337
Insufficient Disk Space	338
Insufficient Processes.....	338
User Problems	338
User Cannot Log In	339
User Cannot Log In Using New Password.....	340
User Cannot Remote Log In to Remote Domain	340
User Cannot Change Password	340
Other NIS+ Problems	341
How to Tell if NIS+ Is Running	341
Replica Update Failure	341
FNS Problems and Solutions	343
Cannot Obtain Initial Context	343
Nothing in Initial Context.....	343
“No Permission” Messages (FNS).....	344
fnlist Does not List Suborganizations.....	345
Cannot Create Host- or User-related Contexts.....	345
Cannot Remove a Context I Created.....	346
“Name in Use” with fnunbind.....	346
“Name in Use” with fncbind/fncreate -s.....	347
fndestroy/fnunbind and “Operation Failed”	347
B. Error Messages	349
About NIS+ and FNS Error Messages.....	349
Error Message Context	349

Context-Sensitive Meanings	350
How Error Messages Are Alphabetized	350
Common NIS+ and FNS Error Messages	351
C. Information in NIS+ Tables	403
Auto_Home Table	404
Auto_Master Table.	405
Bootparams Table.	406
Cred Table.	407
Ethers Table	408
Group Table	409
Hosts Table	410
Mail_aliases Table	411
Netgroup Table.	411
Netmasks Table	413
Networks Table	414
Passwd Table	414
Protocols Table	416
RPC Table	416
Services Table	417
Timezone Table.	418
Index	429

Figures

Figure 2-1	Fully qualified Names of Namespace Components	39
Figure 4-1	An Example Solaris Security Gates and Filters	54
Figure 4-2	Summary of the NIS+ Security Process	58
Figure 4-3	Credentials and Domains	62
Figure 4-4	Authorization Classes	64
Figure 4-5	NIS+ Directory Structure	66
Figure 5-1	keylogin Generates a Principal's Private Key	81
Figure 5-2	Creating the DES Credential	82
Figure 5-3	How nisaddcred Creates a Principal's Keys	90
Figure 7-1	Access Rights Display	119
Figure 7-2	Adding Rights to a Table Entry, Example	129
Figure 7-3	Removing Rights to a Table Entry, Example	130
Figure 16-1	NFS File System—Simple Case	292
Figure 16-2	NFS File System—Multiple Servers	293
Figure A-1	Public Key is Propagated to Directory Objects	325

Tables

Table P-1	Typographic Conventions	xxxii
Table P-2	Shell Prompts	xxxii
Table 1-1	Representation of Wiz Network	6
Table 1-2	NIS Maps	10
Table 1-3	Differences Between NIS and NIS+	14
Table 1-4	NIS+ Namespace Administration Commands	18
Table 2-1	Where NIS+ Files are Stored	22
Table 3-1	NIS+ Tables	45
Table 4-1	NIS+ Security Levels.	59
Table 4-2	Types of Credentials	62
Table 5-1	Secure RPC Netname Format	80
Table 5-2	Where Credential-Related Information Is Stored.	84
Table 5-3	Cred Table Credential Information.	86
Table 5-4	Additional Credential-Related Commands.	88
Table 5-5	Creating Administrator Credentials: Command Summary . .	94
Table 6-1	Re-encrypting Your Private Key : Command Summary	102

Table 6-2	Changing a Root Master's Keys: Command Summary	103
Table 6-3	Remotely Changing Root Master Keys: Command Summary	104
Table 6-4	Changing Keys of a Root Replica	104
Table 6-5	Changing Keys of a Root Replica	105
Table 6-6	<code>nisupdkeys</code> Arguments	106
Table 6-7	Updating a Public Key: Command Summary	107
Table 7-1	Table Entries and Columns	113
Table 7-2	Table, Column, Entry Example 1	114
Table 7-3	Table, Column, Entry Example 2	114
Table 7-4	Table, Column, Entry Example 3	115
Table 7-5	Table, Column, Entry Example 4	115
Table 7-6	Access Rights and Levels and the Objects They Act Upon . . .	116
Table 7-7	Default Access Rights	119
Table 7-8	Access Rights Syntax—Class	121
Table 7-9	Access Rights Syntax—Operator	121
Table 7-10	Access Rights Syntax—Rights	122
Table 7-11	Class, Operator, and Rights Syntax—Examples	122
Table 7-12	Object and Table Entry—Examples	123
Table 7-13	The Seven NIS+ Default Values and <code>nisdefaults</code> Options	124
Table 7-14	Changing Defaults—Examples	127
Table 8-1	Access Rights for <code>passwd</code> Command	148
Table 8-2	Number of Days Since 1/1/70	153
Table 8-3	Related Commands	154
Table 8-4	<i>NIS+ Password Display Format</i>	155
Table 9-1	Commands That Affect Groups	174

Table 9-2	Specifying Group Members and Nonmembers	175
Table 9-3	Rights Required for <code>nisgrpadm</code> Command.....	177
Table 10-1	Options for the <code>nislsls</code> Command	185
Table 10-2	<code>nisrm</code> Syntax Options	192
Table 10-3	Other <code>rpc.nisd</code> Syntax Options.....	193
Table 10-4	Options for the <code>nislog</code> Command	201
Table 10-5	<code>nischttl</code> Syntax Options.....	204
Table 11-1	Columnspec Components	209
Table 11-2	Column Types	210
Table 11-3	<code>niscat</code> Syntax Options.....	216
Table 11-4	Comparison of <code>nisgrep</code> and <code>nismatch</code>	219
Table 11-5	<code>depts.wiz.com</code> . Example Table	219
Table 11-6	Regular Expression Symbols	220
Table 11-7	<code>nismatch</code> and <code>nisgrep</code> Syntax Options	221
Table 11-8	<code>nislsln</code> Syntax Options	223
Table 11-9	<code>nisaddent</code> Syntax Options	227
Table 11-10	Using <code>cat</code> with <code>nisaddent</code>	229
Table 11-11	NIS+ Table Types and Matching NIS Map Names	229
Table 12-1	Possible Switch Sources	235
Table 12-2	Switch Status Messages	236
Table 12-3	Responses to Switch Status Messages	236
Table 12-4	Switch File Comment Examples	238
Table 14-1	<code>fncreate</code> Command Options	258
Table 14-2	<code>fnlookup</code> Command Options	267
Table 14-3	<code>fnlist</code> Command Options.....	269

Table 14-4	<code>fnbind</code> Command Options	273
Table 14-5	<code>fncheck</code> Command Options	279
Table 15-1	NIS+ Root Reference	286
Table 16-1	<code>fncreate_fs</code> Command Options	294
Table A-1	Where Credential-Related Information is Stored	323
Table A-2	Updating a Server's Keys	327
Table C-1	Auto_Home Table	404
Table C-2	Auto_Master Table	405
Table C-3	Bootparams Table	406
Table C-4	Cred Table	408
Table C-5	Ethers Table	408
Table C-6	Group Table	410
Table C-7	Hosts Table	410
Table C-8	Mail_aliases Table	411
Table C-9	Netgroup Table	412
Table C-10	Netmasks Table	413
Table C-11	Networks Table	414
Table C-12	Passwd Table	415
Table C-13	Passwd Table Shadow Column	415
Table C-14	Protocols Table	416
Table C-15	RPC Table	416
Table C-16	Services Table	417
Table C-17	Timezone Table	418

Preface

NIS+ and FNS Administration Guide describes how to customize and administer an existing NIS+ or FNS namespace.

This manual is part of the Solaris™ 2.x System and Network Administration documentation set.

Who Should Use This Book

This book is written primarily for system and network administrators. It assumes the reader is an experienced system administrator. MIS managers can also use this book to evaluate NIS+.

Although this manual introduces concepts relevant to NIS+, it makes no attempt to explain networking fundamentals or to describe the administration tools offered by the Solaris environment. If you administer networks, this manual assumes you already know how they work and have already chosen your favorite tools.

How This Book Is Organized

This manual is organized into four parts:

Part 1 — NIS+ Introduction and Overview

Part 1 gives an overview of name services and describes NIS+:

Chapter 1, “Introduction to Name Services,” describes what name services do and compares DNS, NIS, and NIS+.

Chapter 2, “The NIS+ Namespace,” introduces the Network Information Service Plus namespace.

Chapter 3, “NIS+ Tables and Information,” describes the structure and contents of the NIS+ tables.

Chapter 4, “Security Overview,” describes the NIS+ security system, how it affects the entire NIS+ namespace, and how to administer NIS+ security.

Part 2 — Administering NIS+

Part 2 describes how to administer a functioning NIS+ namespace.

Chapter 5, “Administering NIS+ Credentials,” describes how to use the commands that administer NIS+ credentials.

Chapter 6, “Administering NIS+ Keys,” describes how to use the commands that administer NIS+ keys.

Chapter 7, “Administering NIS+ Access Rights,” describes how to use the commands that administer access rights to NIS+ objects and entries.

Chapter 8, “Administering Passwords,” describes how to use the commands that administer passwords and password aging.

Chapter 9, “Administering NIS+ Groups,” describes how to use the commands that administer NIS+ groups.

Chapter 10, “Administering NIS+ Directories,” describes how to use the commands that administer NIS+ directories.

Chapter 11, “Administering NIS+ Tables,” describes how to use the commands that administer NIS+ tables

Chapter 12, “The Name Service Switch,” describes the software and files that determine which information sources the name services use.

Part 3 — Administering FNS

Part 3 describes how to administer a functioning FNS namespace.

Chapter 14, “Administering FNS in NIS+,” describes how to set up and administer an FNS implementation on top of an NIS+ environment.

Chapter 15, “Federating NIS+ With Global Naming Systems,” describes how to federate NIS+ with DNS and X.500.

Chapter 16, “Administering the File System Namespace,” describes how to create file contexts.

Chapter 17, “Administering the Printer Namespace,” describes how to administer the printer context.

Part 4 — Appendices

Part 4 contains NIS+ appendices and a glossary:

Appendix A, “Problems and Solutions,” describes various types of problems that an NIS+ administrator may encounter and how to solve those problems.

Appendix B, “Error Messages,” provides an alphabetic listing of the most commonly encountered error messages.

Appendix C, “Information in NIS+ Tables,” summarizes the contents of the standard NIS+ tables.

“Glossary,” defines NIS+ and related terms.

Related Books

- *NIS+ and DNS Setup and Configuration Guide*—Describes how to set up, and configure NIS+ and DNS.
- *NIS+ Transition Guide*—Describes how to make the transition from NIS to NIS+.

Additional books not part of the Solaris 2.4 documentation set:

- *DNS and Bind* by Cricket Liu and Paul Albitz, (O’Reilly, 1992).
- *Managing NFS and NIS* by Hal Stern, (O’Reilly, 1993).

What Typographic Changes and Symbols Mean

The following table describes the typographic changes used in this book.

Table P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. machine_name% You have mail.
AaBbCc123	What you type, contrasted with on-screen computer output	machine_name% su Password:
<i>AaBbCc123</i>	Command-line placeholder: replace with a real name or value	To delete a file, type <code>rm filename</code> .
<i>AaBbCc123</i>	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in <i>User's Guide</i> . These are called <i>class</i> options. You <i>must</i> be root to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

Table P-2 Shell Prompts

Shell	Prompt
C shell prompt	machine_name%
C shell superuser prompt	machine_name#
Bourne shell and Korn shell prompt	\$
Bourne shell and Korn shell superuser prompt	#

Part 1—NIS+ Introduction and Overview

This part of the manual focuses on the structure of NIS+.

<i>Introduction to Name Services</i>	<i>page 1</i>
<i>The NIS+ Namespace</i>	<i>page 21</i>
<i>NIS+ Tables and Information</i>	<i>page 45</i>
<i>Security Overview</i>	<i>page 53</i>

Introduction to Name Services



This chapter give's a brief overview describing what *name services* are and what they do. Name services are also called *network information services*. This chapter then briefly describes four such name services: DNS, NIS, FNS, and NIS+. It concludes with a more detailed examination of NIS+.

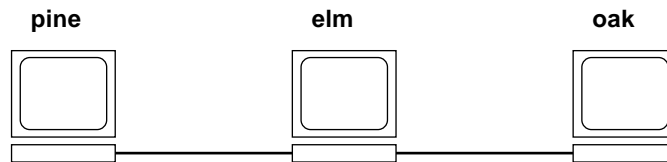
<i>What Is a Name Service?</i>	<i>page 2</i>
<i>DNS</i>	<i>page 8</i>
<i>FNS</i>	<i>page 8</i>
<i>NIS</i>	<i>page 8</i>
<i>NIS+</i>	<i>page 11</i>
<i>What NIS+ Can Do for You</i>	<i>page 12</i>
<i>How NIS+ Differs From NIS</i>	<i>page 13</i>
<i>NIS+ Security</i>	<i>page 16</i>
<i>NIS+ and the Name Service Switch</i>	<i>page 17</i>
<i>Solaris 1.x and NIS-Compatibility Mode</i>	<i>page 17</i>
<i>NIS+ Administration Commands</i>	<i>page 18</i>
<i>NIS+ API</i>	<i>page 20</i>

Directions for setting up NIS+ and DNS namespaces are contained in *NIS+ and DNS Setup and Configuration Guide*. See Glossary for definitions of terms and acronyms you don't recognize.

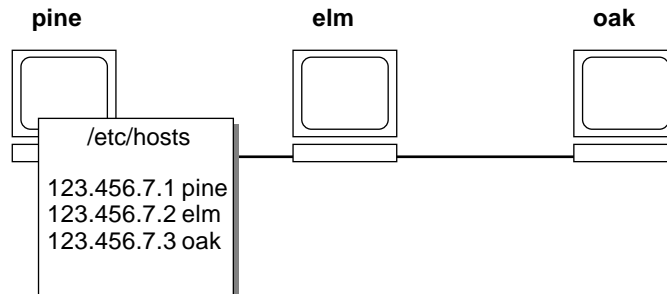
What Is a Name Service?

Name services store information that users, workstations, and applications must have to communicate across the network. Without a name service, each workstation would have to maintain its own copy of this information. This information includes machine addresses, user names, passwords, and network access permissions. The information may be stored in files or database tables. Centrally locating this data makes it easier to administer large networks.

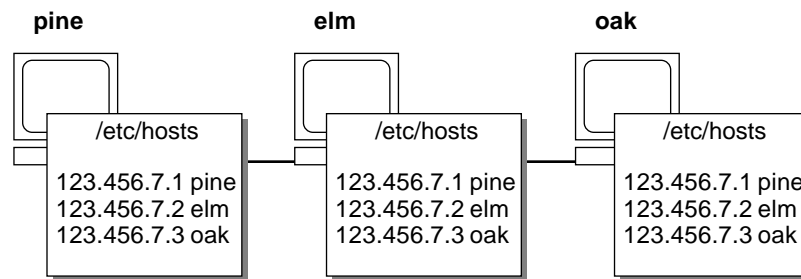
For example, take a simple network of three workstations, `pine`, `elm`, and `oak`:



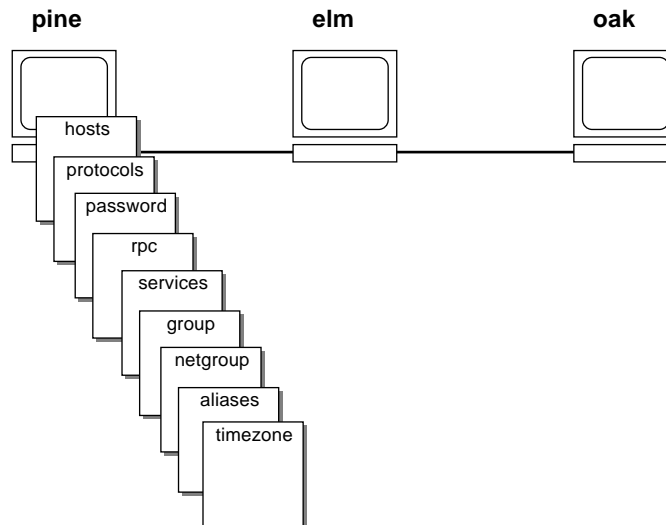
Before `pine` can send a message to either `elm` or `oak`, it must know their network addresses. For this reason, it keeps a file, `/etc/hosts`, that stores the network address of every workstation in the network, including itself.



Likewise, in order for `elm` and `oak` to communicate with `pine` or with each other, they must keep similar files.

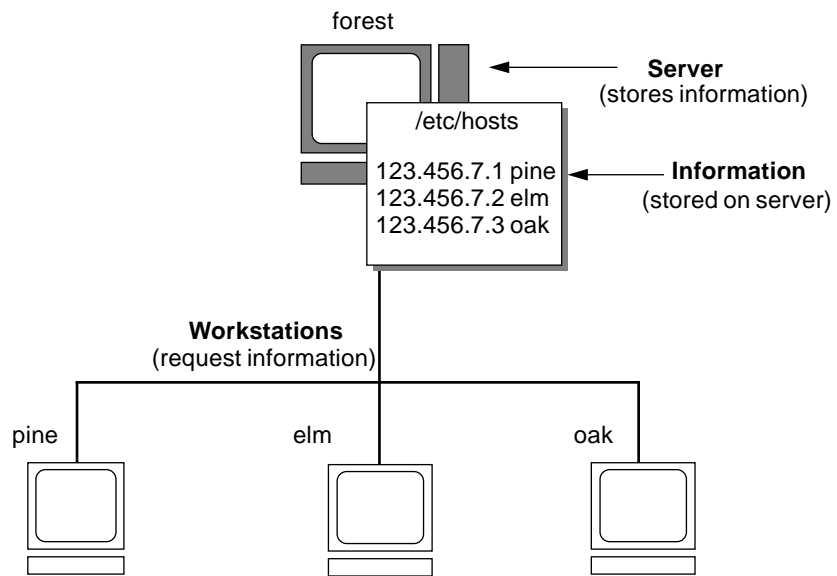


Addresses are not the only network information that workstations need to store. They also need to store security information, mail information, information about their Ethernet interfaces, network services, groups of users allowed to use the network, services offered on the network, and so on. As networks offer more services, the list grows. As a result, each workstation may need to keep an entire set of files similar to `/etc/hosts`:



As this information changes, administrators must keep it current on every workstation in the network. In a small network this is simply tedious, but on a medium or large network, the job becomes not only time-consuming but nearly unmanageable.

A network information service solves this problem. It stores network information on servers and provides it to any workstation that asks for it:

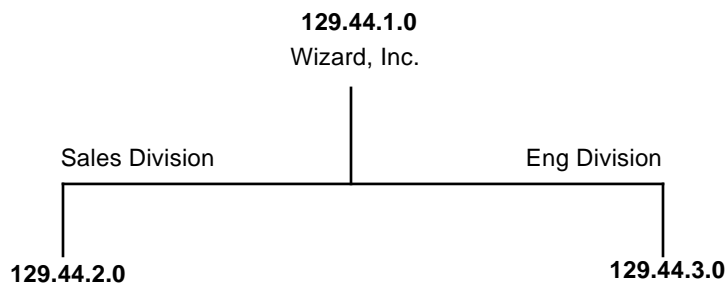


The workstations are known as *clients* of the server. Whenever information about the network changes, instead of updating each client's local file, an administrator updates only the information stored by the network information service. This reduces errors, inconsistencies between clients, and the sheer size of the task.

This arrangement, of a server providing centralized services to clients across a network, is known as *client-server computing*.

Although the chief purpose of a network information service is to centralize information, another is to simplify network names. A network information service enables workstations to be identified by common names instead of numerical addresses. (This is why these services are sometimes called "name services.") This makes communication simpler because users don't have to remember and try to enter cumbersome numerical addresses like "129.44.3.1." Instead, they can use descriptive names like Sales, Lab1, or Arnold.

For example, assume that a fictitious company called Wizard, Inc., has set up a network and connected it to the Internet. The Internet has assigned Wizard, Inc. the network number of 129.44.0.0. Wizard, Inc. has two divisions, Sales and Eng, so its network is divided into two subnets, one for each division. Each subnet has its own address:



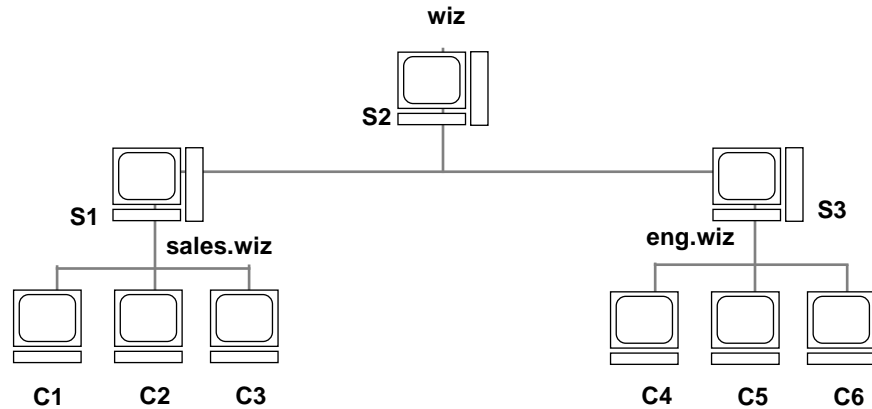
Each division could be identified by its network address, as shown above, but descriptive names made possible by name services would be preferable:



So, instead of addressing mail or other network communications to 129.44.1.0, they could be addressed simply to Wiz. Instead of addressing them to 129.44.2.0 or 129.44.3.0, they could be addressed to sales.wiz or eng.wiz.

Names are also more flexible than physical addresses. While physical networks tend to remain stable, the organizations that use them tend to change. A network information service can act as a buffer between an organization and its physical network. This is because a network information service is mapped to the physical network, not hard-wired to it. For example,

assume that the Wiz network is supported by three servers, S1, S2, and S3, and that two of those servers, S1 and S3, support clients:

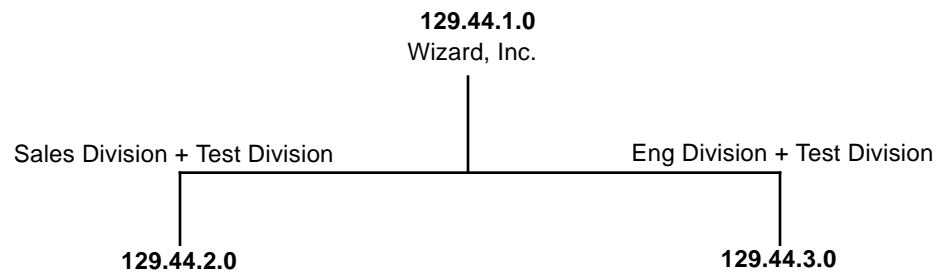


Clients C1, C2, and C3 would obtain their network information from server S1. Clients C4, C5, and C6 would obtain it from server S3. The resulting network is summarized in Table 1-1. (The table is a generalized representation of that network but does not resemble an actual network information map.)

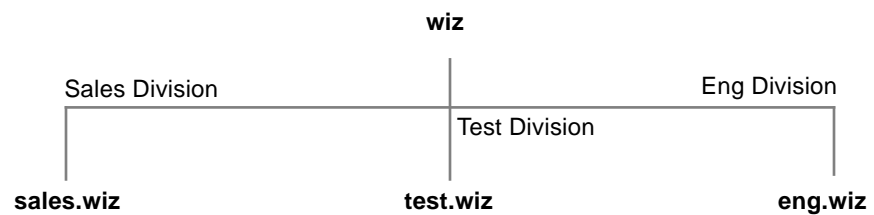
Table 1-1 Representation of Wiz Network

Network Address	Network Name	Server	Clients
129.44.1.0	wiz	S1	
129.44.2.0	sales.wiz	S2	C1, C2, C3
129.44.3.0	eng.wiz	S3	C4, C5, C6

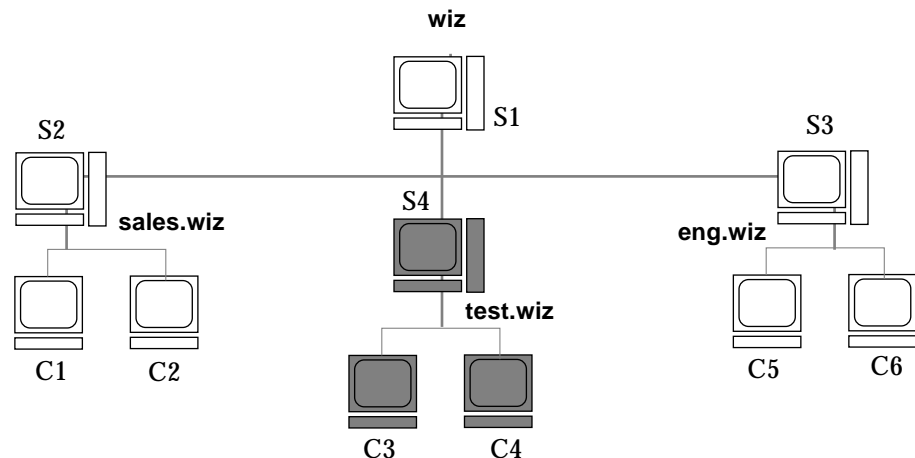
Now assume that Wizard, Inc. created a third division, Testing, which borrowed some resources from the other two divisions, but did not create a third subnet. The physical network would then no longer parallel the corporate structure:



Traffic for the Test Division would not have its own subnet, but would instead be split between 129.44.2.0 and 129.44.3.0. However, with a network information service, the Test Division traffic could have its own dedicated network:



Thus, when an organization changes, its network information service can simply change its mapping:



Now clients C1 and C2 would obtain their information from server S2; C3 and C4 from server S4; and C5 and C6 from server S3.

Subsequent changes in the Wizard Inc., organization would continue to be accommodated by changes to the “soft” network information structure without reorganizing the “hard” network structure.

DNS

DNS, the Domain Name System, is the name service provided by the Internet for TCP/IP networks. It was developed so that workstations on the network could be identified with common names instead of Internet addresses. DNS performs naming between hosts *within* your local administrative domain and *across* domain boundaries.

The collection of networked workstations that use DNS are referred to as the DNS *namespace*. The DNS namespace can be divided into a hierarchy of domains. A DNS domain is simply a group of workstations. Each domain is supported by two or more name servers: a principal server and one or more secondary servers. Each server implements DNS by running a daemon called `in.named`. On the client's side, DNS is implemented through the “resolver.” The resolver's function is to resolve users' queries; to do that, it queries a name server, which then returns either the requested information or a referral to another server.

FNS

FNS, the Federated Naming Service, supports the use of different autonomous naming systems in a single Solaris environment. FNS allows you to use a single, simple naming system interface for all of the different name services on your network.

NIS

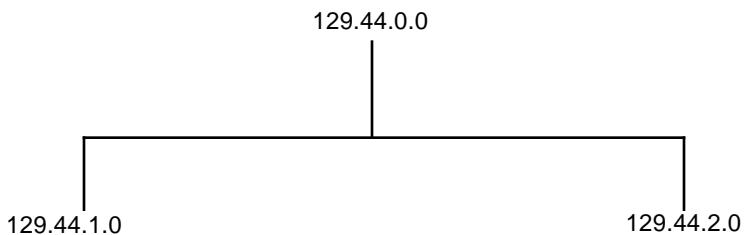
NIS, a network information service, was developed independently of DNS and had a slightly different focus. Whereas DNS focused on making communication simpler by using workstation names instead of addresses, NIS focused on making network administration more manageable by providing centralized control over a variety of network information. As a result, NIS

stores information not only about workstation names and addresses, but also about users, the network itself, and network services. This collection of network information is referred to as the NIS *namespace*.

NIS Architecture

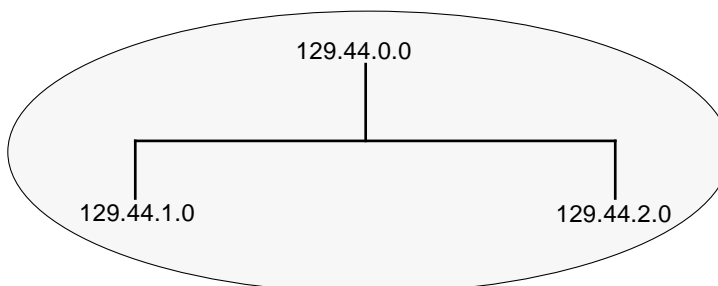
NIS uses a client-server arrangement similar to DNS. Replicated NIS servers provide services to NIS clients. The principal servers are called *master* servers, and for reliability, they have backup, or *replica* servers (sometimes referred to as *slave* server). Both master and replica servers use the NIS information retrieval software and both store NIS maps.

NIS, like DNS, uses domains to arrange the workstations, users, and networks in its namespace. However, it does not use a domain hierarchy; an NIS namespace is flat. Thus, this physical network:



would be arranged into one NIS domain:

The wiz Domain



An NIS domain can't be connected directly to the Internet. However, organizations that want to use NIS and be connected to the Internet can combine NIS with DNS. They use NIS to manage all local information and

DNS for host name resolution. NIS provides special client routines for this purpose (“DNS forwarding”). When a client needs access to any type of information except IP addresses, the request goes to the client’s NIS server. When a client needs name resolution, the request goes to the DNS server. From the DNS server, the client has access to the Internet in the usual way.

NIS Maps

NIS stores information in a set of files called maps. NIS maps were designed to replace UNIX[®] /etc files, as well as other configuration files, so they store much more than names and addresses. As a result, the NIS namespace has a large set of maps, as shown in Table 1-2.

NIS maps are essentially two-column tables. One column is the key and the other column is information about the key. NIS finds information for a client by searching through the keys. Thus, some information is stored in several maps because each map uses a different key. For example, the names and addresses of workstations are stored in two maps: `hosts.byname` and `hosts.byaddr`. When a server has a workstation’s name and needs to find its address, it looks in the `hosts.byname` map. When it has the address and needs to find the name, it looks in the `hosts.byaddr` map.

Table 1-2 NIS Maps

NIS Map	Description
<code>bootparams</code>	Lists the names of the diskless clients and the location of the files they need during booting
<code>ethers.byaddr</code>	Lists the Ethernet addresses of workstations and their corresponding names
<code>ethers.byname</code>	Lists the names of workstations and their corresponding Ethernet addresses
<code>group.bygid</code>	Provides membership information about groups, using the group id as the key
<code>group.byname</code>	Provides membership information about groups, using the group name as the key
<code>hosts.byaddr</code>	Lists the names and addresses of workstations, using the address as the key
<code>hosts.byname</code>	Lists the names and addresses of workstations, using the name as the key

Table 1-2 NIS Maps (Continued)

NIS Map	Description
<code>mail.aliases</code>	Lists the mail aliases in the namespace and all the workstations that belong to them
<code>mail.byaddr</code>	Lists the mail aliases in the namespace, using the address as the key
<code>netgroup</code>	Contains netgroup information, using group name as the key
<code>netgroup.byhost</code>	Contains information about the netgroups in the namespace, using workstation names as the key
<code>netgroup.byuser</code>	Contains netgroup information, using user as the key
<code>netid.byname</code>	Contains the Secure RPC netname of workstations and users, along with their UIDs and GIDs
<code>netmasks.byaddr</code>	Contains network masks used with IP subnetting, using address as the key
<code>networks.byaddr</code>	Contains the names and addresses of the networks in the namespace, and their Internet addresses
<code>networks.byname</code>	Contains the names and addresses of the networks in the namespace, using the names as the key
<code>passwd.byname</code>	Contains password information, with username as the key
<code>passwd.byuid</code>	Contains password information, with userid as the key
<code>protocols.byname</code>	Lists the network protocols used
<code>protocols.bynumber</code>	Lists the network protocols used but uses their number as the key
<code>publickey.byname</code>	Contains public and secret keys for Secure RPC
<code>rpc.bynumber</code>	Lists the known program name and number of RPCs
<code>services.byname</code>	Lists the available Internet services
<code>ypservers</code>	Lists the NIS servers in the namespace, along with their IP addresses

NIS+

NIS+ is a network name service similar to NIS but with more features. NIS+ is not an extension of NIS. It is a new software program.

NIS+ enables you to store information about workstation addresses, security information, mail information, Ethernet interfaces, and network services in central locations where all workstations on a network can have access to it. This configuration of network information is referred to as the NIS+ *namespace*.

The NIS+ namespace is hierarchical, and is similar in structure to the UNIX directory file system. The hierarchical structure allows an NIS+ namespace to be configured to conform to the logical hierarchy of an organization. The namespace's layout of information is unrelated to its *physical* arrangement. Thus, an NIS+ namespace can be divided into multiple domains that can be administered autonomously. Clients may have access to information in other domains in addition to their own if they have the appropriate permissions.

NIS+ uses a client-server model to store and have access to the information contained in an NIS+ namespace. Each domain is supported by a set of servers. The principal server is called the *master* server and the backup servers are called *replicas*. The network information is stored in 16 standard NIS+ tables in an internal NIS+ database. Both master and replica servers run NIS+ server software and both maintain copies of NIS+ tables. Changes made to the NIS+ data on the master server are incrementally propagated automatically to the replicas.

NIS+ includes a sophisticated security system to protect the structure of the namespace and its information. It uses authentication and authorization to verify whether a client's request for information should be fulfilled. *Authentication* determines whether the information requester is a valid user on the network. *Authorization* determines whether a particular user is allowed to have or modify the information requested.

Solaris clients use the name service switch (`/etc/nsswitch.conf` file) to determine from where a workstation will retrieve network information. Such information may be stored in local `/etc` files, NIS, DNS, or NIS+. You can specify different sources for different types of information in the name service switch.

What NIS+ Can Do for You

NIS+ has some major advantages over NIS:

- Secure data access
- Hierarchical and decentralized network administration

- Very large namespace administration
- Access to resources across domains
- Incremental updates

With the security system described in “NIS+ Security” on page 16, you can control a particular user’s access to an individual entry in a particular table. This approach to security helps to keep the system secure and administration tasks to be more broadly distributed without risking damage to the entire NIS+ namespace or even to an entire table.

The NIS+ hierarchical structure allows for multiple domains in one namespace. Division into domains makes administration easier to manage. Individual domains can be administered completely independently, thereby relieving the burden on system administrators who would otherwise each be responsible for very large namespaces. As mentioned above, the security system in combination with decentralized network administration allows for a sharing of administrative work load.

Even though domains may be administered independently, all clients can be granted permission to access information across all domains in a namespace. Since a client can only see the tables in its own domain, the client can only have access to tables in other domains by explicitly addressing them.

Incremental updates mean faster updates of information in the namespace. Since domains are administered independently, changes to master server tables only have to be propagated to that master’s replicas and not to the entire namespace. Once propagated, these updates are visible to the entire namespace immediately.

How NIS+ Differs From NIS

NIS+ differs from NIS in several ways. It has many new features and the terminology for similar concepts is different. Look in the Glossary if you see a term you don’t recognize. Table 1-3 gives an overview of the major differences between NIS and NIS+.

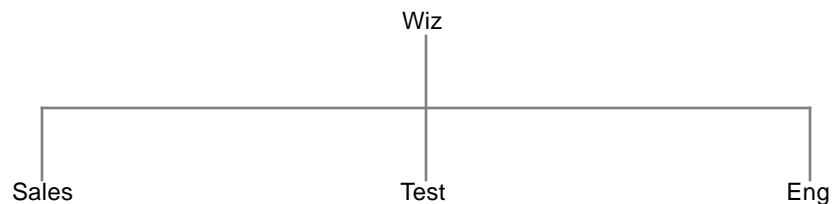
Table 1-3 Differences Between NIS and NIS+

NIS	NIS+
Flat domains—no hierarchy	Hierarchical layout—data stored in different levels in the namespace
Data stored in two column maps	Data stored in multicolumn tables
Uses no authentication	Uses DES authentication
Single choice of network information source	Name service switch—lets client choose information source: NIS, NIS+, DNS, or local <code>/etc</code> files
Updates delayed for batch propagation	Incremental updates propagated immediately

NIS+ was designed to replace NIS. NIS addresses the administration requirements of client-server computing networks prevalent in the 1980s. At that time client-server networks did not usually have more than a few hundred clients and a few multipurpose servers. They were spread across only a few remote sites, and since users were sophisticated and trusted, they did not require security.

However, client-server networks have grown tremendously since the mid-1980s. They now range from 100-10,000 multivendor clients supported by 10-100 specialized servers located in sites throughout the world, and they are connected to several “untrusted” public networks. In addition, the information they store changes much more rapidly than it did during the time of NIS. The size and complexity of these networks required new, autonomous administration practices. NIS+ was designed to address these requirements.

The NIS namespace, being flat, centralizes administration. Because networks in the 1990s require scalability and decentralized administration, the NIS+ namespace was designed with hierarchical domains, like those of DNS:



This design enables NIS+ to be used in a range of networks, from small to very large. It also allows the NIS+ service to adapt to the growth of an organization. For example, if a corporation splits itself into two divisions, its NIS+ namespace could be divided into two domains that could be administered autonomously. Just as the Internet delegates administration of domains *downward*, NIS+ domains can be administered more or less independently of each other.

Although NIS+ uses a domain hierarchy similar to that of DNS, an NIS+ domain is much more than a DNS domain. A DNS domain only stores name and address information about its clients. An NIS+ domain, on the other hand, is a collection of *information* about the workstations, users, and network services in a portion of an organization.

Although this division into domains makes administration more autonomous and growth easier to manage, it does not make information harder to access. Clients have the same access to information in other domains as they would have had under one umbrella domain. A domain can even be administered from within another domain.

The principal NIS+ server is called the *master* server, and the backup servers are called *replicas*. Both master and replica servers run NIS+ server software and both maintain copies of NIS+ tables. Tables store information in NIS+ the way maps store information in NIS. The principal server stores the original tables, and the backup servers store copies.

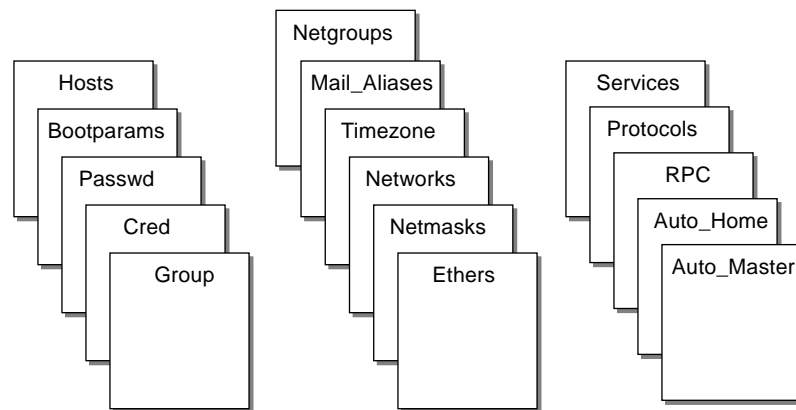
However, NIS+ uses an updating model that is completely different from the one used by NIS. Since at the time NIS was developed, the type of information it would store changed infrequently, NIS was developed with an update model that focused on stability. Its updates are handled manually and, in large organizations, can take more than a day to propagate to all the replicas. Part of the reason for this is the need to remake and propagate an entire map every time any information in the map changes.

NIS+, however, accepts *incremental* updates. Changes must still be made on the master server, but once made they are automatically propagated to the replica servers and immediately made available to the entire namespace. You don't have to "make" any maps or wait for propagation.

Details about NIS+ domain structure, servers, and clients, are provided in Chapter 2, "The NIS+ Namespace."

An NIS+ domain can be connected to the Internet through its NIS+ clients, using the name service switch (see “NIS+ and the Name Service Switch” on page 17). The client, if it is also a DNS client, can set up its switch configuration file to search for information in either DNS zone files or NIS maps—in addition to NIS+ tables.

NIS+ stores information in *tables* instead of maps or zone files. NIS+ provides 16 types of predefined, or *system*, tables:



Each table stores a different type of information. For instance, the hosts table stores information about workstation addresses, while the passwd table stores information about users of the network.

NIS+ tables provide two major improvements over the maps used by NIS. First, an NIS+ table can be searched by any column, not just the first column (sometimes referred to as the “key”). This eliminates the need for duplicate maps, such as the `hosts.byname` and `hosts.byaddr` maps used by NIS. Second, the information in NIS+ tables can be accessed and manipulated at three levels of granularity: the table level, the entry level, and the column level. NIS+ tables—and the information stored in them—are described in Chapter 3, “NIS+ Tables and Information.”

NIS+ Security

NIS+ protects the structure of the namespace, and the information it stores, by the complementary processes of *authorization* and *authentication*.

- *Authorization.* Every component in the namespace specifies the type of operation it will accept and from whom. This is authorization.
- *Authentication.* NIS+ attempts to *authenticate* every request for access to the namespace. Requests come from NIS+ *principals*. A NIS+ principal can be a process, machine, root, or a user. Valid NIS+ principals possess a NIS+ *credential*. NIS+ authenticates the originator of the request (principal) by checking the principal's credential.

If the principal possesses an authentic (valid) credential, and if the principal's request is one that the principal is authorized to perform, NIS+ carries out the request. If either the credential is missing or invalid, or the request is not one the principal is authorized to perform, NIS+ denies the request for access. A much fuller description of the entire NIS+ security system is provided in Chapter 4, "Security Overview." "

NIS+ and the Name Service Switch

NIS+ works in conjunction with a separate program called the *name service switch*. The name service switch, sometimes referred to as "the switch," enables Solaris 2.x-based workstations to obtain their information from more than one name service; specifically, from local or `/etc` files, NIS maps, DNS zone files, or NIS+ tables. The switch not only offers a choice of sources, but allows a workstation to specify different sources for different *types* of information. A complete description of the switch software and its associated files is provided in Chapter 12, "The Name Service Switch."

Solaris 1.x and NIS-Compatibility Mode

Although NIS+ is provided with the 2.x package, it can be used by workstations running NIS on the Solaris 1.x software. To access NIS+ service on machines running Solaris 1.x, you must run the NIS+ servers in *NIS-compatibility mode*.

NIS-compatibility mode enables an NIS+ server running Solaris 2.x to answer requests from NIS clients while continuing to answer requests from NIS+ clients. NIS+ does this by providing two service interfaces. One responds to NIS+ client requests, while the other responds to NIS client requests.

This mode does not require any additional setup or changes to NIS clients. In fact, NIS clients are not even aware that the server that is responding isn't an NIS server — except for some differences including: the NIS+ server running in NIS-compatibility mode does not support the `ypupdate` and `ypxfr` protocols and thus it cannot be used as a replica or master NIS server. For more information on NIS-compatibility mode, see *NIS+ Transition Guide*.

Note – In Solaris 2.3 and later releases, the NIS-compatibility mode *supports* DNS-forwarding. In Solaris 2.2, support for DNS forwarding is available as a *patch*. The DNS forwarding patch is *not* available in Solaris 2.0 and 2.1 releases.

Two more differences need to be pointed out. One is that instructions for setting up a server in NIS-compatibility mode are slightly different than those used to set up a standard NIS+ server. For details, see *NIS+ and DNS Setup and Configuration Guide*. The other is that NIS-compatibility mode has security implications for tables in the NIS+ namespace. Since the NIS client software does not have the capability to provide the credentials that NIS+ servers expect from NIS+ clients, all their requests end up classified as *unauthenticated*. Therefore, to allow NIS clients to access information in NIS+ tables, those tables must provide access rights to unauthenticated requests. This is handled automatically by the utilities used to set up a server in NIS-compatibility mode, as described in Part 2. However, to understand more about the authentication process and NIS-compatibility mode, see Chapter 4, “Security Overview.”

NIS+ Administration Commands

NIS+ provides a full set of commands for administering a namespace. Table 1-4, below, summarizes them.

Table 1-4 NIS+ Namespace Administration Commands

Command	Description
<code>nisaddcred</code>	Creates credentials for NIS+ principals and stores them in the cred table.
<code>nisaddent</code>	Adds information from <code>/etc</code> files or NIS maps into NIS+ tables.
<code>nis_cachemgr</code>	Starts the NIS+ cache manager on an NIS+ client.
<code>niscat</code>	Displays the contents of NIS+ tables.

Table 1-4 NIS+ Namespace Administration Commands (Continued)

Command	Description
<code>nischgrp</code>	Changes the group owner of an NIS+ object.
<code>nischmod</code>	Changes an object's access rights.
<code>nischown</code>	Changes the owner of an NIS+ object.
<code>nischttl</code>	Changes an NIS+ object's time-to-live value.
<code>nisdefaults</code>	Lists an NIS+ object's default values: domain name, group name, workstation name, NIS+ principal name, access rights, directory search path, and time-to-live.
<code>nisgrep</code>	Searches for entries in an NIS+ table.
<code>nisgrpadm</code>	Creates or destroys an NIS+ group, or displays a list of its members. Also adds members to a group, removes them, or tests them for membership in the group.
<code>nisinit</code>	Initializes an NIS+ client or server.
<code>nisln</code>	Creates a symbolic link between two NIS+ objects.
<code>nisls</code>	Lists the contents of an NIS+ directory.
<code>nismatch</code>	Searches for entries in an NIS+ table.
<code>nismkdir</code>	Creates an NIS+ directory and specifies its master and replica servers.
<code>nispasswd</code>	Changes password information stored in the NIS+ passwd table. (Rather than using <code>nispasswd</code> , you should use <code>passwd</code> or <code>passwd -r nisplus</code> .)
<code>nisrm</code>	Removes NIS+ objects (except directories) from the namespace.
<code>nisrmdir</code>	Removes NIS+ directories and replicas from the namespace.
<code>nissetup</code>	Creates <code>org_dir</code> and <code>groups_dir</code> directories and a complete set of (unpopulated) NIS+ tables for an NIS+ domain.
<code>nisshowcache</code>	Lists the contents of the NIS+ shared cache maintained by the NIS+ cache manager.

Table 1-4 NIS+ Namespace Administration Commands (Continued)

Command	Description
<code>nistbladm</code>	Creates or deletes NIS+ tables, and adds, modifies or deletes entries in an NIS+ table.
<code>nisupdkeys</code>	Updates the public keys stored in an NIS+ object.
<code>passwd</code>	Changes password information stored in the NIS+ Passwd table. Also administers password aging and other password-related parameters.

NIS+ API

The NIS+ application programmer's interface (API) is a group of functions that can be called by an application to access and modify NIS+ objects. The NIS+ API has 54 functions that fall into nine categories:

- Object manipulation functions (`nis_names`)
- Table access functions (`nis_tables`)
- Local name functions (`nis_local_names`)
- Group manipulation functions (`nis_groups`)
- Application subroutine functions (`nis_subr`)
- Miscellaneous functions (`nis_misc`)
- Database access functions (`nis_db`)
- Error message display functions (`nis_error`)
- Transaction log functions (`nis_admin`)

These category names match the names by which they are grouped in the NIS+ man pages.

The NIS+ Namespace



The NIS+ name service is designed to conform to the shape of the organization that installs it, wrapping itself around the bulges and corners of almost any network configuration. This is implemented through the NIS+ namespace. This chapter describes the structure of the NIS+ namespace, the servers that support it, and the clients that use it.

<i>NIS+ Files and Directories</i>	<i>page 22</i>
<i>Structure of the NIS+ Namespace</i>	<i>page 22</i>
<i>Directories</i>	<i>page 24</i>
<i>Domains</i>	<i>page 25</i>
<i>Servers</i>	<i>page 27</i>
<i>NIS+ Clients and Principals</i>	<i>page 30</i>
<i>Naming Conventions</i>	<i>page 37</i>
<i>NIS+ Name Expansion</i>	<i>page 43</i>

NIS+ Files and Directories

Table 2-1 lists the directories used to store NIS+ files.

Table 2-1 Where NIS+ Files are Stored

Directory	Where	Contains
/usr/bin	All machines	NIS+ user commands
/usr/lib/nis	All machines	NIS+ administrator commands
/usr/sbin	All machines	NIS+ daemons
/usr/lib/	All machines	NIS+ shared libraries
/var/nis/data	NIS+ server	Data files used by NIS+ server
/var/nis	NIS+ server	NIS+ working files
/var/nis	NIS+ client machines	Machine-specific data files used by NIS+



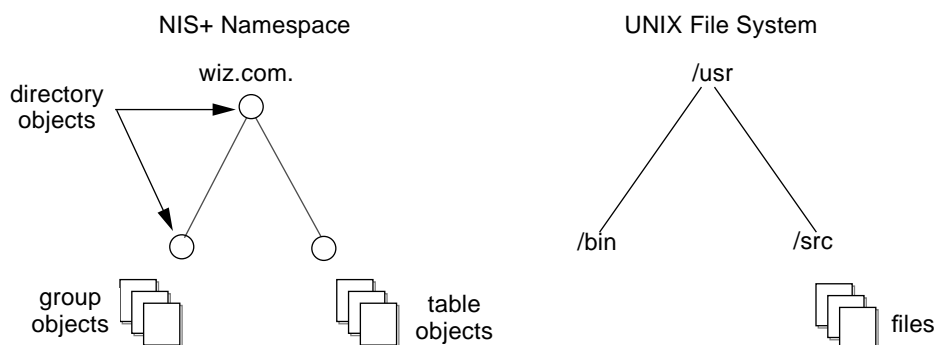
Caution – Do not rename the `/var/nis` or `/var/nis/data` directories or any of the files in these directories that were created by `nisinit` or any of the other NIS+ setup procedures. In Solaris 2.4 and earlier, the `/var/nis` directory contained two files named `hostname.dict` and `hostname.log`. It also contained a subdirectory named `/var/nis/hostname`. In Solaris 2.5, the two files are named `trans.log` and `data.dict`, and the subdirectory is named `/var/nis/data`. In Solaris 2.5, the *content* of the files has also been changed and they are not backward compatible with Solaris 2.4 or earlier. Thus, if you rename either the directories or the files to match the Solaris 2.4 patterns, the files will not work with either the Solaris 2.4 or the Solaris 2.5 version of `rpc.nisd`. Therefore, you should not rename either the directories or the files.

Structure of the NIS+ Namespace

The NIS+ namespace is the arrangement of information stored by NIS+. The namespace can be arranged in a variety of ways to suit the needs of an organization. For example, if an organization had three divisions, its NIS+ namespace would likely be divided into three parts, one for each division. Each part would store information about the users, workstations, and network

services in its division, but the parts could easily communicate with each other. Such an arrangement would make information easier for the users to access and for the administrators to maintain.

Although the arrangement of an NIS+ namespace can vary from site to site, all sites use the same structural components: directories, tables, and groups. These components are called NIS+ *objects*. NIS+ objects can be arranged into a hierarchy that resembles a UNIX file system. For example, the illustration below shows, on the left, a namespace that consists of three directory objects, three group objects, and three table objects; on the right it shows a UNIX file system that consists of three directories and three files:



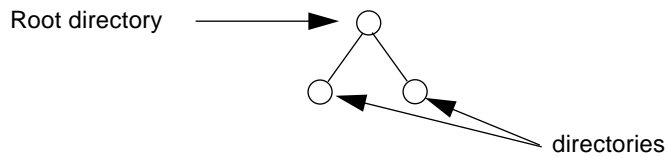
Although an NIS+ namespace resembles a UNIX file system, it has five important differences:

- Although both use directories, the other objects in an NIS+ namespace are tables and groups, not files.
- The NIS+ namespace is administered only through NIS+ administration commands (listed in Table 1-4 on page 18) or graphical user interfaces designed for that purpose, such as the Solstice™ AdminSuite™ tools; it cannot be administered with standard UNIX file system commands or GUIs.
- The names of UNIX file system components are separated by slashes (/usr/bin), but the names of NIS+ namespace objects are separated by dots (wiz.com.).
- The “root” of a UNIX file system is reached by stepping through directories from right to left (/usr/src/file1), while the root of the NIS+ namespace is reached by stepping from left to right (sales.wiz.com.).

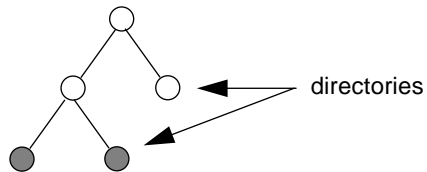
- Because NIS+ object names are structured from left to right, a fully qualified name always ends in a dot. Any NIS+ object ending in a dot is assumed to be a fully qualified name. NIS+ object names that do not end in a dot are assumed to be relative names.

Directories

Directory objects are the skeleton of the namespace. When arranged into a tree-like structure, they divide the namespace into separate parts. You may want to visualize a directory hierarchy as an upside-down tree, with the root of the tree at the top and the leaves toward the bottom. The topmost directory in a namespace is the *root* directory. If a namespace is flat, it has only one directory, but that directory is nevertheless the root directory. The directory objects beneath the root directory are simply called “directories”:

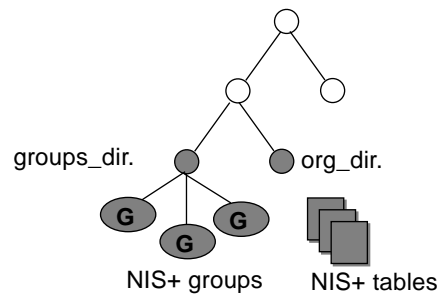


A namespace can have several levels of directories:



When identifying the relation of one directory to another, the directory beneath is called the *child* directory and the directory above is called the *parent* directory.

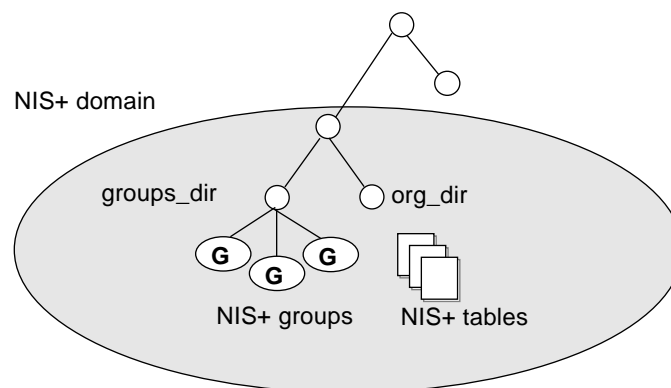
Whereas UNIX directories are designed to hold UNIX files, NIS+ directories are designed to hold NIS+ objects: other directories, tables and groups. Any NIS+ directory that stores NIS+ groups is named `groups_dir`. Any directory that stores NIS+ system tables is named `org_dir`.



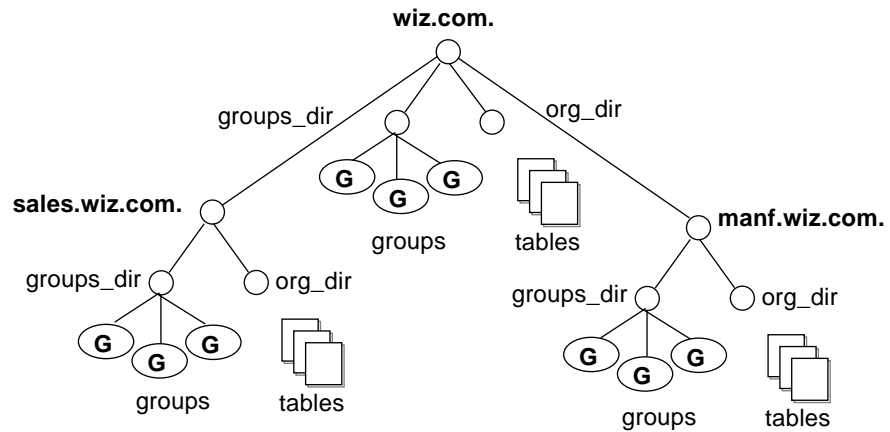
Technically, you can arrange directories, tables, and groups into any structure that you like. However, NIS+ directories, tables, and groups in a namespace are normally arranged into configurations called *domains*. Domains are designed to support separate portions of the namespace. For instance, one domain may support the Sales Division of a company, while another may support the Manufacturing Division.

Domains

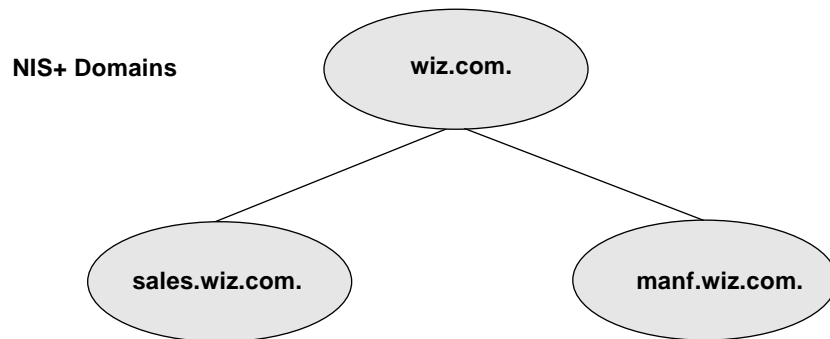
An NIS+ domain consists of a directory object, its `org_dir` directory, its `groups_dir` directory, and a set of NIS+ tables.



NIS+ domains are not *tangible* components of the namespace. They are simply a convenient way to *refer* to sections of the namespace that are used to support real-world organizations. For example, assume that the Wizard Corporation has a Sales division and an Manufacturing division. To support those divisions, its NIS+ namespace would most likely be arranged into three major directory groups, with a structure that looked like this:



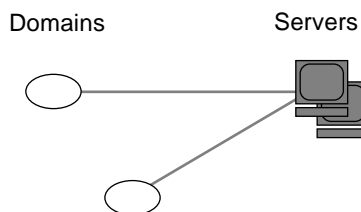
Instead of referring to such a structure as three directories, six subdirectories, and several additional objects, referring to it as three domains is more convenient:



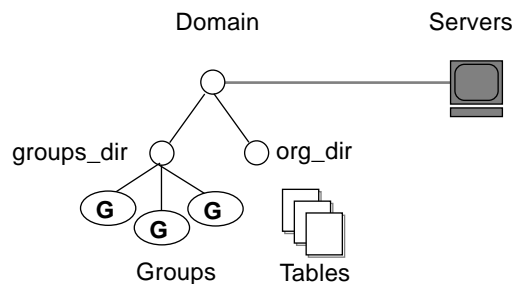
Part 2 of this manual describes how to configure domains.

Servers

Every NIS+ domain is supported by a set of NIS+ servers. The servers store the domain's directories, groups, and tables, and answer requests for access from users, administrators, and applications. Each domain is supported by only one set of servers. However, a single set of servers can support more than one domain:



Remember that a domain is not an object but only refers to a collection of objects. Therefore, a server that supports a domain is not actually associated with the domain, but with the domain's main directory:



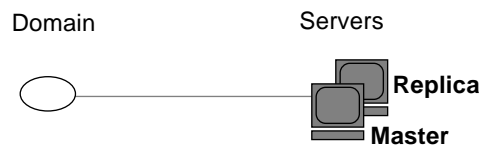
This connection between the server and the directory object is established during the process of setting up a domain. Although instructions are provided in Part 2, one thing is important to mention now: when that connection is established, the directory object stores the name and IP address of its server. This information is used by clients to send requests for service, as described later in this section.

Any Solaris 2.x-based workstation can be an NIS+ server. The software for both NIS+ servers and clients is bundled together into the release. Therefore, any workstation that has the Solaris 2.x software installed can become a server or a

client, or both. What distinguishes a client from a server is the *role* it is playing. If a workstation is providing NIS+ service, it is acting as an NIS+ server. If it is requesting NIS+ service, it is acting as an NIS+ client.

Because of the need to service many client requests, a workstation that will act as an NIS+ server might be configured with more computing power and more memory than the average client. And, because it needs to store NIS+ data, it might also have a larger disk. However, other than hardware to improve its performance, a server is not inherently different from an NIS+ client.

Two types of servers support an NIS+ domain: a master and its replicas:



The master server of the root domain is called the *root master* server. A namespace has only one root master server. The master servers of other domains are simply called master servers. Likewise, there are root replica servers and regular replica servers.

Both master and replica servers store NIS+ tables and answer client requests. The master, however, stores the master copy of a domain's tables. The replicas store only duplicates. The administrator loads information into the tables in the master server, and the master server propagates it to the replica servers.

This arrangement has two benefits. First, it avoids conflicts between tables because only one set of master tables exists; the tables stored by the replicas are only copies of the masters. Second, it makes the NIS+ service much more available. If either the master or a replica is down, another server can act as a backup and handle the requests for service.

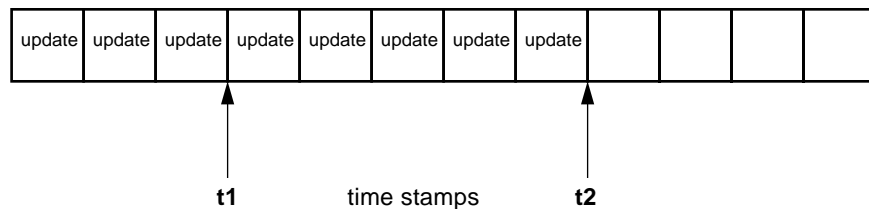
How Servers Propagate Changes

An NIS+ master server implements updates to its objects immediately; however, it tries to “batch” several updates together before it propagates them to its replicas. When a master server receives an update to an object, whether a directory, group, link, or table, it waits about two minutes for any other

updates that may arrive. Once it is finished waiting, it stores the updates in two locations: on disk and in a *transaction log* (it has already stored the updates in memory).

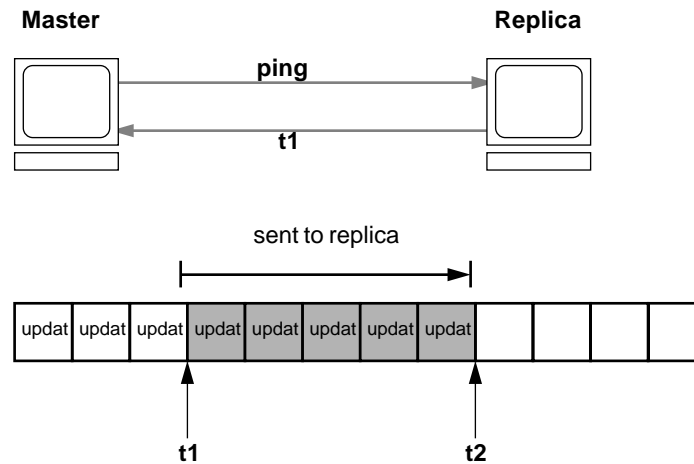
The transaction log is used by a master server to store changes to the namespace until they can be propagated to replicas. A transaction log has two primary components: updates and time stamps.

Transaction Log



An update is an actual copy of a changed object. For instance, if a directory has been changed, the update is a complete copy of the directory object. If a table entry has been changed, the update is a copy of the actual table entry. The time stamp indicates the time at which an update was made by the master server.

After recording the change in the transaction log, the master sends a message to its replicas, telling them that it has updates to send them. Each replica replies with the time stamp of the last update it received from the master. The master then sends each replica the updates it has recorded in the log since the replica's time stamp:



When the master server updates all its replicas, it clears the transaction log. In some cases, such as when a new replica is added to a domain, the master receives a time stamp from a replica that is before its earliest time stamp still recorded in the transaction log. If that happens, the master server performs a full *resynchronization*, or *resync*. A resync downloads all the objects and information stored in the master down to the replica. During a resync, both the master and replica are busy. The replica cannot answer requests for information; the master can answer read requests but cannot accept update requests. Both respond to requests with a `Server Busy - Try Again` or similar message.

NIS+ Clients and Principals

NIS+ principals are the entities (clients) that submit requests for NIS+ services.

Principal

An NIS+ principal may be someone who is logged in to a client machine as a regular user or someone who is logged in as superuser (root). In the first instance, the request actually comes from the client user; in the second instance, the request comes from the client workstation. Therefore, an NIS+ principal can be a client user or a client workstation.

(An NIS+ principal can also be the entity that supplies an NIS+ service from an NIS+ server. Since all NIS+ servers are also NIS+ clients, much of this discussion also applies to servers.)

Client

An NIS+ client is a workstation that has been set up to receive NIS+ service. Setting up an NIS+ client consists of establishing security credentials, making it a member of the proper NIS+ groups, verifying its home domain, verifying its switch configuration file and, finally, running the NIS+ initialization script. (Complete instructions are provided in Part 2.)

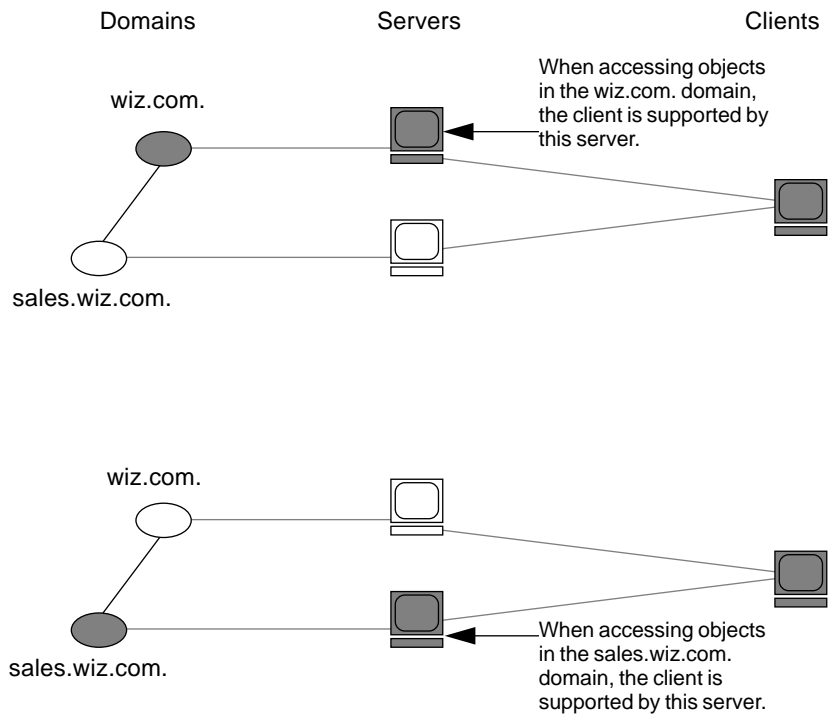
An NIS+ client can access any part of the namespace, subject to security constraints. In other words, if it has been authenticated and has been granted the proper permissions, it can access information or objects in any domain in the namespace.

Although a client can access the entire namespace, a client belongs to only one domain, which is referred to as its *home* domain. A client's home domain is usually specified during installation, but it can be changed or specified later. All the information about a client, such as its IP address and its credentials, is stored in the NIS+ tables of its home domain.

There is a subtle difference between being an NIS+ client and being listed in an NIS+ table. Entering information about a workstation into an NIS+ table does not automatically make that workstation an NIS+ client. It simply makes information about that workstation available to all NIS+ clients. That workstation cannot request NIS+ service unless it is actually set up as an NIS+ client.

Conversely, making a workstation an NIS+ client does not enter information about that workstation into an NIS+ table. It simply allows that workstation to receive NIS+ service. If information about that workstation is not explicitly entered into the NIS+ tables by an administrator, other NIS+ clients will not be able to get it.

When a client requests access to the namespace, it is actually requesting access to a particular domain in the namespace. Therefore, it sends its request to the server that supports the domain it is trying to access. Here is a simplified representation:

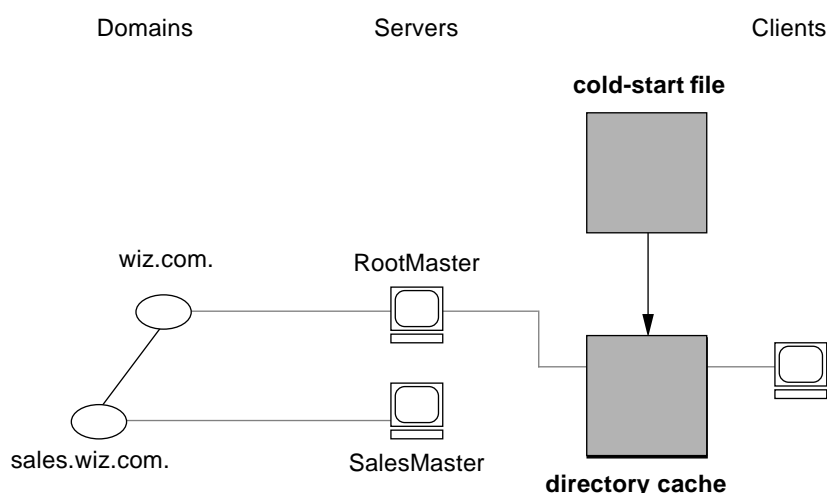


How does the client know which server that is? By trial and error. Beginning with its home server, the client tries first one server, then another, until it finds the right one. When a server cannot answer the client's request, it sends the client information to help locate the right server. Over time, the client builds up its own cache of information and becomes more efficient at locating the right server. The next section describes this process.

The Cold-Start File and Directory Cache

When a client is initialized, it is given a *cold-start file*. The cold-start file gives a client a copy of a directory object that it can use as a starting point for contacting servers in the namespace. The directory object contains the address, public keys, and other information about the master and replica servers that support the directory. Normally, the cold-start file contains the directory object of the client's home domain.

A cold-start file is used only to initialize a client's *directory cache*. The directory cache, managed by an NIS+ facility called the *cache manager*, stores the directory objects that enable a client to send its requests to the proper servers.



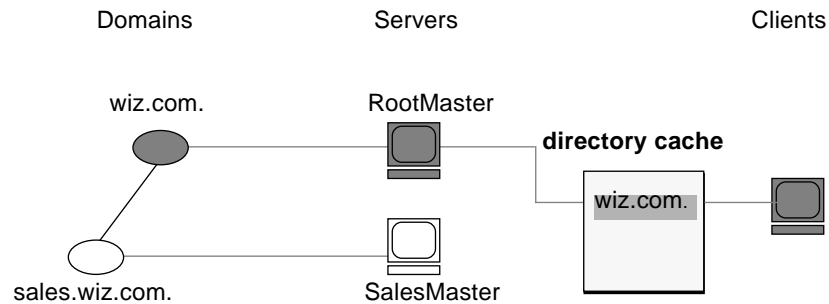
By storing a copy of the namespace's directory objects in its directory cache, a client can know which servers support which domains. (To view the contents of a client's cache, use the `nisshowcache` command, described in "The `nisshowcache` Command" on page 197.) Here is a simplified example:

Domain	Directory Name	Supporting Server	IP Address
wiz.com.	wiz.com.	RootMaster	129.44.1.1
sales.wiz.com	sales.wiz.com.	SalesMaster	129.44.2.1
manf.wiz.com.	manf.wiz.com.	ManfMaster	129.44.3.1
int.sales.wiz.com.	int.sales.wiz.com.	IntlSalesMaster	129.44.2.11

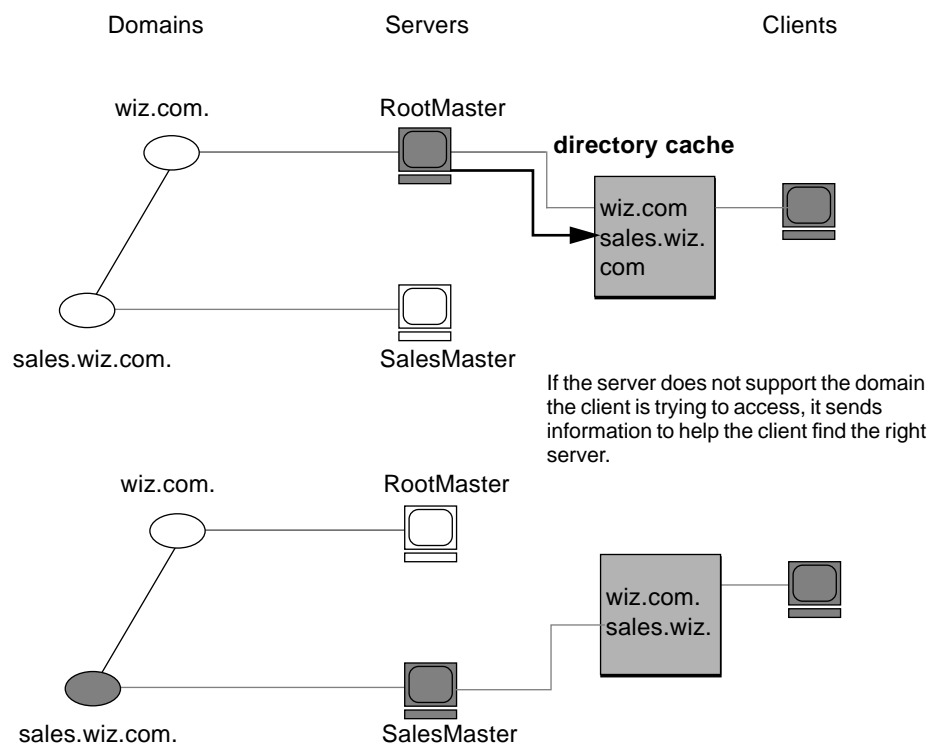
To keep these copies up-to-date, each directory object has a *time-to-live* (TTL) field. Its default value is 12 hours. If a client looks in its directory cache for a directory object and finds that it has not been updated in the last 12 hours, the cache manager obtains a new copy of the object. You can change a directory object's time-to-live value with the `nischttl` command, as described in "The `nischttl` Command" on page 203. However, keep in mind that the longer the

time-to-live, the higher the likelihood that the copy of the object will be out of date; and the shorter the time to live, the greater the network traffic and server load.

How does the directory cache accumulate these directory objects? As mentioned above, the cold-start file provides the first entry in the cache. Therefore, when the client sends its first request, it sends the request to the server specified by the cold-start file. If the request is for access to the domain supported by that server, the server answers the request.



If the request is for access to another domain (for example, sales.wiz.com.), the server tries to help the client locate the proper server. If the server has an entry for that domain in its own directory cache, it sends a copy of the domain's directory object to the client. The client loads that information into its directory cache for future reference and sends its request to that server.



In the unlikely event that the server does not have a copy of the directory object the client is trying to access, it sends the client a copy of the directory object for its own home domain, which lists the address of the server's parent. The client repeats the process with the parent server, and keeps trying until it finds the proper server or until it has tried all the servers in the namespace. What the client does after trying all the servers in the domain is determined by the instructions in its name service switch configuration file. See Chapter 12, "The Name Service Switch," for details.

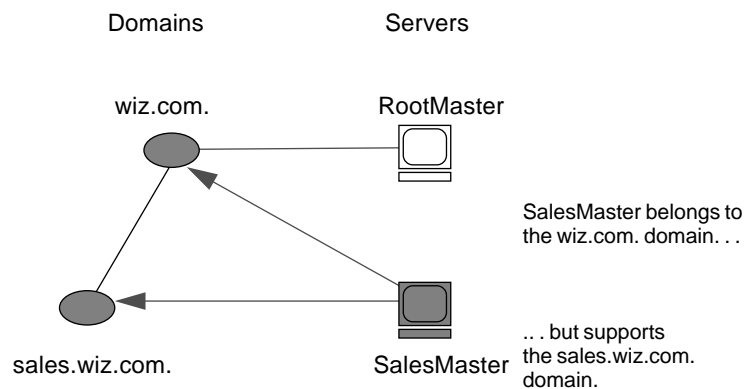
Over time, the client accumulates in its cache a copy of all the directory objects in the namespace and thus the IP addresses of the servers that support them. When it needs to send a request for access to another domain, it can usually find the name of its server in its directory cache and send the request directly to that server.

An NIS+ Server Is Also a Client

An NIS+ server is also an NIS+ client. In fact, before you can set up a workstation as a server (as described in Part 2), you must initialize it as a client. The only exception is the root master server, which has its own unique setup process.

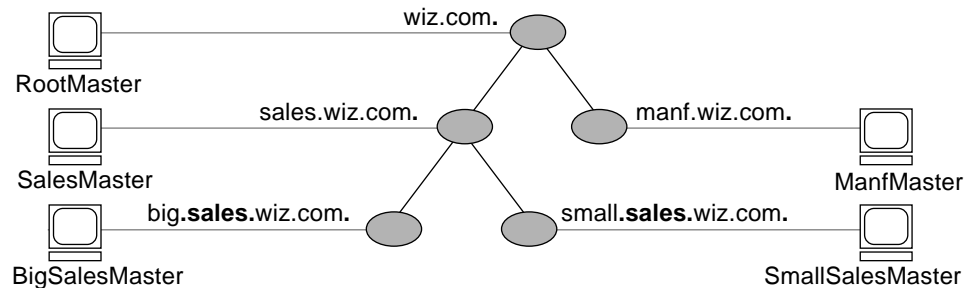
This means that in addition to *supporting* a domain, a server also *belongs* to a domain. In other words, by virtue of being a client, a server has a home domain. Its host information is stored in the Hosts table of its home domain, and its DES credentials are stored in the cred table of its home domain. Like other clients, it sends its requests for service to the servers listed in its directory cache.

An important point to remember is that—except for the root domain—a server’s home domain is the parent of the domain the server supports:



In other words, a server supports clients in one domain, but is a client of another domain. A server cannot be a client of a domain that it supports, with the exception of the root domain. Because they have no parent domain, the servers that support the root domain belong to the root domain itself.

For example, consider the following namespace:



The chart lists which domain each server supports and which domain it belongs to:

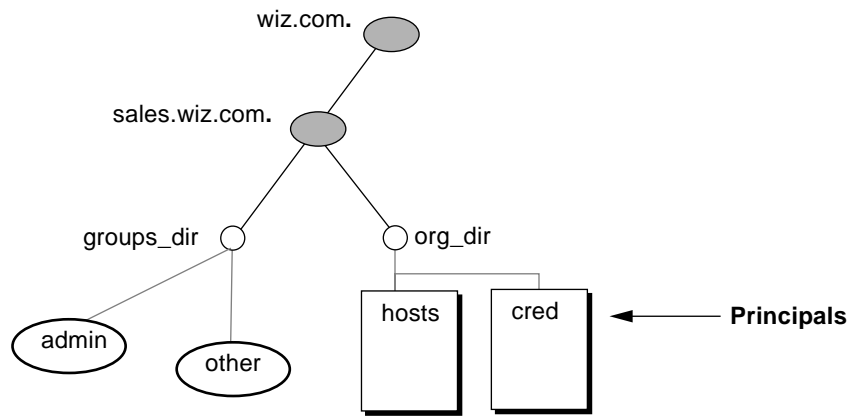
Server	Supports	Belongs to
RootMaster	wiz.com.	wiz.com.
SalesMaster	sales.wiz.com.	wiz.com.
BigSalesMaster	big.sales.wiz.com.	sales.wiz.com.
SmallSalesMaster	small.sales.wiz.com.	sales.wiz.com.
ManfMaster	manf.wiz.com.	wiz.com.

Naming Conventions

Objects in an NIS+ namespace can be identified with two types of names: *partially qualified* and *fully qualified*. A partially qualified name, also called a *simple* name, is simply the name of the object or any portion of the fully qualified name. If during any administration operation you type the partially qualified name of an object or principal, NIS+ will attempt to expand the name into its fully qualified version. For details, see "NIS+ Name Expansion" on page 43.

A fully qualified name is the complete name of the object, including all the information necessary to locate it in the namespace, such as its parent directory, if it has one, and its complete domain name, including a trailing dot.

This varies among different types of objects, so the conventions for each type, as well as for NIS+ principals, is described separately. This namespace will be used as an example:



The fully qualified names for all the objects in this namespace, including NIS+ principals, are summarized in Figure 2-1.

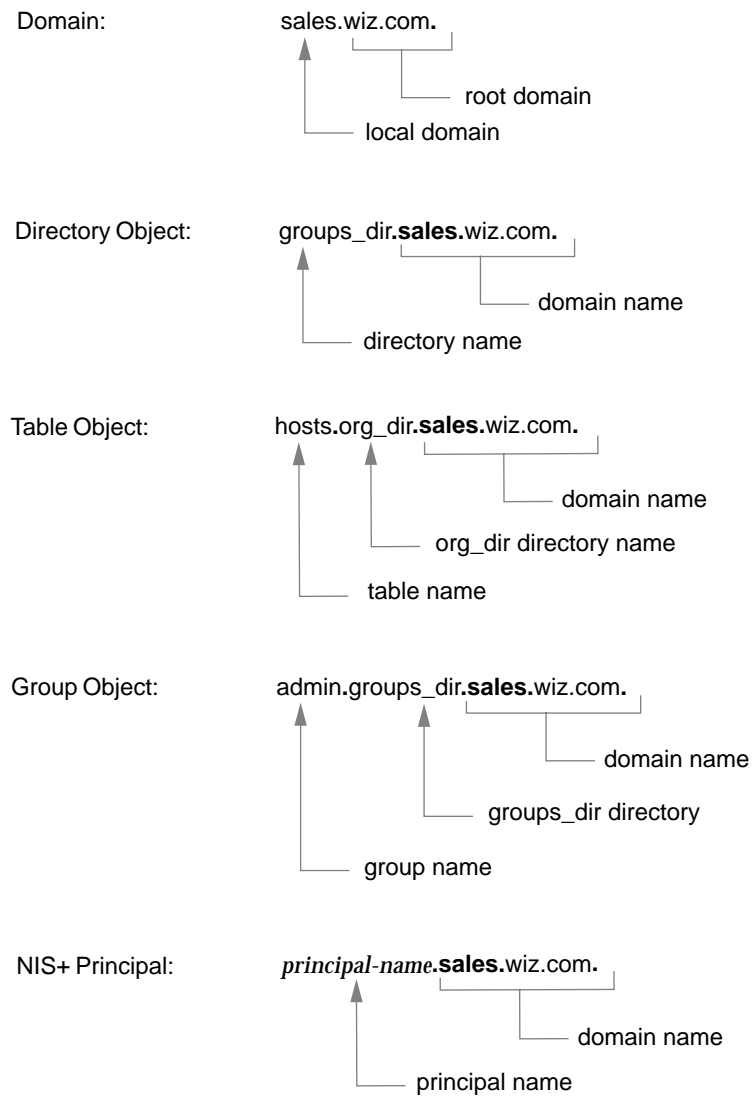


Figure 2-1 Fully qualified Names of Namespace Components

NIS+ Domain Names

A fully qualified NIS+ domain name is formed from left to right, starting with the local domain and ending with the root domain:

wiz.com.

sales.wiz.com.

intl.sales.wiz.com.

The first line shows the name of the root domain. The root domain must always have at least two labels and must end in a dot. The second label can be an Internet domain name, such as `com`. The second and third lines show the names of lower-level domains.

Directory Object Names

A directory's simple name is simply the name of the directory object. Its fully qualified name consists of its simple name plus the fully qualified name of its domain (which always includes a trailing dot):

groups_dir (simple name)

groups_dir.manf.wiz.com. (fully qualified name)

If you set up an unusual hierarchy in which several layers of directories do not form a domain, be sure to include the names of the intermediate directories. For example:

lowest_dir.lower_dir.low_dir.mydomain.com.

The simple name is normally used from within the same domain, and the fully qualified name is normally used from a remote domain. However, by specifying search paths in a domain's `NIS_PATH` environment variable, you can use the simple name from remote domains (see "NIS+ Name Expansion" on page 43).

Tables and Group Names

Fully qualified table and group names are formed by starting with the object name and appending the directory name, followed by the fully qualified domain name. Remember that all system table objects are stored in an

`org_dir` directory and all group objects are stored in a `groups_dir` directory. (If you create your own NIS+ tables, you can store them anywhere you like.) Here are some examples of group and table names:

<code>admin.groups_dir.wiz.Inc.</code>	<code>admin.groups_dir.wiz.com.</code>
<code>admin.groups_dir.sales.wiz.Inc.</code>	<code>admin.groups_dir.sales.wiz.com.</code>
<code>hosts.org_dir.wiz.Inc.</code>	<code>hosts.org_dir.wiz.com.</code>
<code>hosts.org_dir.sales.wiz.Inc.</code>	<code>hosts.org_dir.sales.wiz.com.</code>

Table Entry Names

To identify an entry in an NIS+ table, you need to identify the table object and the entry within it. This type of name is called an *indexed* name. It has the following syntax:

[*column=value* , *column=value* , . . .] , *table-name*

Column is the name of the table column. *Value* is the actual value of that column. *Table-name* is the fully qualified name of the table object. Here are a few examples of entries in the hosts table:

```
[addr=129.44.2.1,name=pine],hosts.org_dir.sales.wiz.com.  
[addr=129.44.2.2,name=elm],hosts.org_dir.sales.wiz.com.  
[addr=129.44.2.3,name=oak],hosts.org_dir.sales.wiz.com.
```

You can use as few column-value pairs inside the brackets as required to uniquely identify the table entry.

Some NIS+ administrative commands accept variations on this syntax. For details, see the `nistbladm`, `nismatch`, and `nisgrep` commands in Part 2.

Host Names

Host names may contain up to 24 characters. Letters, numbers, the dash (-) and underscore (_) characters are allowed in host names. Host names are not case sensitive (that is, upper and lower case letters are treated as the same). The first character of a host name must be a letter of the alphabet. Blank spaces are not permitted in host names.

Note – Dots (.) are not permitted in host names. For example, a host name such as `myco.2` is not permitted. Dots are not allowed in host names even if they are enclosed in quotes. For example, `'myco.2'` is not permitted. Dot are

only used as part of a fully qualified host name to identify the domain components. For example, `myco-2.sales.wiz.com` is a correct fully qualified host name.

Domains and hosts should not have the same name. For example, if you have a sales domain you should not have a machine named `sales`. Similarly, if you have a machine named `home`, you do not want to create a domain named `home`. This caution applies to subdomains, for example if you have a machine named `west` you don't want to create a `sales.west.myco.com` subdomain.

NIS+ Principal Names

NIS+ principal names are sometimes confused with Secure RPC netnames. Both types of names are described in the security chapters of Part—2. However, one difference is worth pointing out now because it can cause confusion: NIS+ principal names *always* end in a dot and Secure RPC netnames *never* do:

```

olivia.sales.wiz.com.      (NIS+ principal name)
unix.olivia@sales.wiz.com (Secure RPC netname)

```

Also, even though credentials for principals are stored in a cred table, neither the name of the cred table nor the name of the `org_dir` directory is included in the principal name.

Accepted Name Symbols

You can form namespace names from any printable character in the ISO Latin 1 set. However, the names cannot start with these characters:

```

@      < >      +      [ ]      -      /
=      .      ,      :      ;

```

To use a string, enclose it in double quotes. To use a quote sign in the name, quote the sign too (for example, to use `o'henry`, type `o'"henry`). To include white space (as in John Smith), use double quotes within single quotes, like this:

```


`"John Smith"`


```

See “Host Names” on page 41 for restrictions that apply to host names.

NIS+ Name Expansion

Entering fully qualified names with your NIS+ commands can quickly become tedious. To ease the task, NIS+ provides a name-expansion facility. When you enter a partially qualified name, NIS+ attempts to find the object by looking for it under different directories. It starts by looking in the default domain. This is the home domain of the client from which you type the command. If it does not find the object in the default domain, NIS+ searches through each of the default domain's parent directories in ascending order until it finds the object. It stops after reaching a name with only two labels. Here are some examples (assume you are logged onto a client that belongs to the `software.big.sales.wiz.com.` domain).

`mydir`  *expands into* `mydir.Software.big.sales.wiz.com.`
`mydir.big.sales.wiz.com.`
`mydir.sales.wiz.com.`
`mydir.wiz.com.`

`hosts.org_dir`  *expands into* `hosts.org_dir Software.big.sales.wiz.com.`
`hosts.org_dir.big.sales.wiz.com.`
`hosts.org_dir.sales.wiz.com.`
`hosts.org_dir.wiz.com.`

NIS_PATH Environment Variable

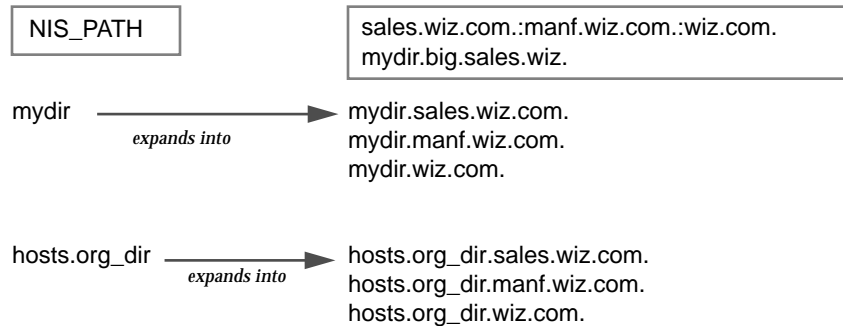
You can change or augment the list of directories NIS+ searches through by changing the value of the environment variable `NIS_PATH`. `NIS_PATH` accepts a list of directory names separated by colons:

```
setenv NIS_PATH directory1:directory2:directory3...
```

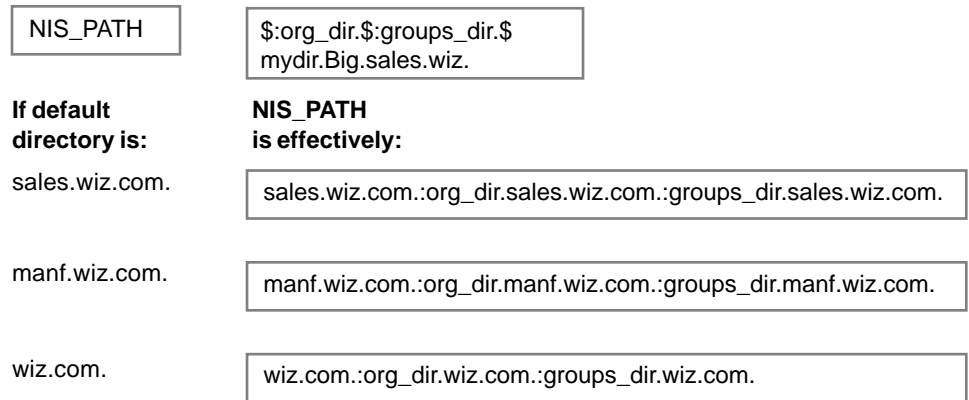
or

```
NIS_PATH=directory1:directory2:directory3...;export NIS_PATH
```

NIS+ searches through these directories from left to right. For example:



Like \$PATH and \$MANPATH, the NIS_PATH variable accepts the special symbol, \$. You can append the \$ symbol to a directory name or add it by itself. If you append it to a directory name, NIS+ appends the default directory to that name. For example:



If you use the \$ sign by itself (for example, org_dir.\$:\$), NIS+ performs the standard name expansion described earlier: it starts looking in the default directory and proceeds through the parent directories. In other words, the default value of NIS_PATH is \$.

Note - Keep in mind that additions and changes to your NIS_PATH may increase the number of lookups that NIS+ has to perform and thus slow down performance.

NIS+ Tables and Information



NIS+ stores a wide variety of network information in tables. This chapter describes the structure of those tables and provides a brief overview of how they can be set up

<i>NIS+ Table Structure</i>	<i>page 45</i>
<i>Ways to Set Up Tables</i>	<i>page 50</i>

NIS+ Table Structure

NIS+ tables provide several features not found in simple text files or maps. They have a column-entry structure, they accept search paths, they can be linked together, and they can be set up in several different ways. NIS+ provides 16 preconfigured system tables, and you can also create your own tables. Table 3-1 lists the preconfigured NIS+ tables.

Table 3-1 NIS+ Tables

Table	Information in the Table
hosts	Network address and host name of every workstation in the domain
bootparams	Location of the root, swap, and dump partition of every diskless client in the domain
passwd	Password information about every user in the domain.
cred	Credentials for principals who belong to the domain

Table 3-1 NIS+ Tables (Continued)

Table	Information in the Table
group	The group name, group password, group ID, and members of every UNIX group in the domain
netgroup	The netgroups to which workstations and users in the domain may belong
mail_aliases	Information about the mail aliases of users in the domain
timezone	The time zone of every workstation in the domain
networks	The networks in the domain and their canonical names
netmasks	The networks in the domain and their associated netmasks
ethers	The Ethernet address of every workstation in the domain
services	The names of IP services used in the domain and their port numbers
protocols	The list of IP protocols used in the domain
RPC	The RPC program numbers for RPC services available in the domain
auto_home	The location of all user's home directories in the domain
auto_master	Automounter map information

These tables store a wide variety of information, ranging from user names to Internet services. Most of this information is generated during a setup or configuration procedure. For instance, an entry in the passwd table is created when a user account is set up. An entry in the hosts table is created when a workstation is added to the network. And an entry in the networks table is created when a new network is set up.

Since this information is generated from such a wide field of operations, much of it is beyond the scope of this manual. However, as a convenience, Appendix C, "Information in NIS+ Tables," summarizes the information contained in each column of the tables, providing details only when necessary to keep things from getting confusing, such as when distinguishing groups from NIS+ groups and netgroups. For thorough explanations of the information, consult Solaris 2.x system and network administration manuals.

The Cred table, because it contains only information related to NIS+ security, is described in Chapter 5, "Administering NIS+ Credentials."

Columns and Entries

Although NIS+ tables store different types of information, they all have the same underlying structure; they are each made up of rows and columns (the rows are called “entries” or “entry objects”):

Column

Entry

A client can access information by a key, or by any column that is searchable. For example, to find the network address of a workstation named `baseball`, a client could look through the `hostname` column until it found `baseball`.

**Hostname
Column**

	nose		
	grass		
	violin		
	baseball		

it then would move along the `baseball` entry to find its network address:

	Address Column	Hostname Column		
		nose		
		grass		
		violin		
Baseball Row	129.44.1.2	baseball		
	←			

Because a client can access table information at any level, NIS+ provides security mechanisms for all three levels. For instance, an administrator could assign read rights to everyone for a table at the object level, modify rights to the owner at the column level, and modify rights to the group at the entry level. Details about table security are provided in Chapter 7, “Administering NIS+ Access Rights.”

Search Paths

A table contains information only about its *local* domain. For instance, tables in the *wiz.com.* domain contain information only about the users, clients, and services of the *wiz.com.* domain. The tables in the *sales.wiz.com.* domain store information only about the users, clients, and services of the *sales.wiz.com.* domain. And so on.

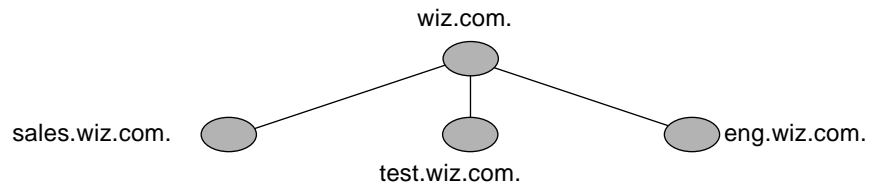
If a client in one domain tries to find information that is stored in another domain, it has to provide a fully qualified name. As described in “NIS+ Name Expansion” on page 43” if the `NIS_PATH` environment variable is set up properly, the NIS+ service will do this automatically.

Every NIS+ table can also specify a *search path* that a server will follow when looking for information. The search path is an ordered list of NIS+ tables, separated by colons:

```
table: table: table. . .
```

The table names in the search path don't have to be fully qualified; they can be expanded just like names entered at the command line. When a server cannot find information in its local table, it returns the table's search path to the client. The client uses that path to look for the information in every table named in the search path, in order, until it finds the information or runs out of names.

Here is an example that demonstrates the benefit of search paths. Assume the following domain hierarchy:



The hosts tables of the lower three domains have the following contents:

sales.wiz.com.		test.wiz.com.		eng.wiz.com.	
127.0.0.1	localhost	127.0.0.1	localhost	127.0.0.1	localhost
129.44.2.10	vermont	129.44.4.10	nebraska	129.44.3.10	georgia
129.44.2.11	maine	129.44.4.11	oklahoma	129.44.3.11	florida
129.44.2.12	cherry	129.44.4.12	corn	129.44.3.12	orange
129.44.2.13	apple	129.44.4.13	wheat	129.44.3.13	potato
129.44.2.14	mailhost	129.44.4.14	mailhost	129.44.3.14	mailhost

Assume now that a user logged onto a client in the sales.wiz.com. domain wants to log in remotely to another client. If that user does not provide a fully qualified name, it can only remotely log on to five workstations: `vermont`, `maine`, `cherry`, `apple`, and the `mailhost`.

Now assume that the search path of the hosts table in the sales.wiz.com. domain listed the hosts tables from the test.wiz.com. and eng.wiz.com. domains:

```
hosts.org_dir.test.wiz.com.:hosts.org_dir.eng.wiz.com.
```

Now a user in the `sales.wiz.com.` domain can enter something like `rlogin oklahoma`, and the NIS+ server will find it. It will first look for `oklahoma` in the local domain, but when it does not find a match, it will look in the `test.wiz.com.` domain. How does the client know how to find the `test.wiz.com.` domain? As described in Chapter 2, “The NIS+ Namespace,” the information is stored in its directory cache. If it is not stored in its directory cache, the client will obtain the information by following the process described in Chapter 2, “The NIS+ Namespace”.

There is a slight drawback, though, to specifying a search path. If the user were to enter an incorrect name, such as `rlogin potatoe`, the server would need to look through three tables—instead of just one—before returning an error message. If you set up search paths throughout the namespace, an operation may end up searching through the tables in 10 domains instead of just 2 or 3. Another drawback is a performance loss from having many clients contact more than one set of servers when they need to access NIS+ tables.

You should also be aware that since “mailhost” is often used as an alias, when trying to find information about a specific mailhost, you should use its fully qualified name (for example, `mailhost.sales.wiz.com.`), or NIS+ will return *all* the mailhosts it finds in all the domains it searches through.

You can specify a table’s search path by using the `-p` option to the `nistbladm` command, as described in “The `nistbladm` Command” on page 208.

Ways to Set Up Tables

Setting up NIS+ tables involves three or four tasks:

1. Creating the `org_dir` directory
2. Creating the system tables
3. Creating nonsystem tables (optional)
4. Populating the tables with information

As described in Chapter 2, “The NIS+ Namespace,” NIS+ system tables are stored under an `org_dir` directory. So, before you can create any tables, you must create the `org_dir` directory that will hold them. You can do this in three ways.

- Use the `nisserver` script. The `nisserver` script creates the appropriate directories and a full set of system tables. Running the `nisserver` script is the recommended method.
- Use the `nismkdir` command. The `nismkdir` command simply creates the directory.
- Use the `/usr/lib/nis/nissetup` utility. The `nissetup` utility creates the `org_dir` and `groups_dir` directories and a full set of system tables.

The `nisserver` script and the `nissetup` and `nismkdir` utilities are described in *NIS+ and DNS Setup and Configuration Guide*. Additional information on the `nismkdir` command can be found in “The `nismkdir` Command” on page 187.

A benefit of the `nissetup` utility is its capability to assign the proper access rights to the tables of a domain whose servers are running in NIS-compatibility mode. When entered with the `-Y` flag, it assigns read permissions to the `nobody` class of the objects it creates, allowing NIS clients, who are unauthenticated, to get information from the domain’s NIS+ tables.

The 16 NIS+ system tables and the type of information they store are described in Appendix C, “Information in NIS+ Tables.” To create them, you could use one of the three ways mentioned above. The `nistbladm` utility also creates and modifies NIS+ tables. You could conceivably create all the tables in a namespace with the `nistbladm` command, but you would have to type much more and you would have to know the correct column names and access rights. A much, much easier way is to use the `nisserver` script.

To create a nonsystem table—that is, a table that has not been preconfigured by NIS+—use the `nistbladm` command.

You can populate NIS+ tables in three ways: from NIS maps, from ASCII files (such as `/etc` files), and manually.

If you are upgrading from the NIS service, you already have most of your network information stored in NIS maps. You don’t have to re-enter this information manually into NIS+ tables. You can transfer it automatically with the `nispopulate` script or the `nisaddent` utility.

If you are not using another network information service, but maintain network data in a set of `/etc` files, you don’t have to re-enter this information, either. You can transfer it automatically, also using the `nispopulate` script or the `nisaddent` utility.

If you are setting up a network for the first time, you may not have much network information stored anywhere. In that case, you'll need to first get the information and then enter it manually into the NIS+ tables. You can do this with the `nistbladm` command. You can also do it by entering all the information for a particular table into an *input file*—which is essentially the same as an `/etc` file—and then transferring the contents of the file with the `nispopulate` script or the `nisaddent` utility.

How Tables Are Updated

When a domain is set up, its servers receive their first versions of the domain's NIS+ tables. These versions are stored on disk, but when a server begins operating, it loads them into memory. When a server receives an update to a table, it immediately updates its memory-based version of the table. When it receives a request for information, it uses the memory-based copy for its reply.

Of course, the server also needs to store its updates on disk. Since updating disk-based tables takes time, all NIS+ servers keep log files for their tables. The log files are designed to temporarily store changes made to the table, until they can be updated on disk. They use the table name as the prefix and append `.log`. For example:

```
hosts.org_dir.log
bootparams.org_dir.log
password.org_dir.log
```

You should update disk-based copies of a table on a daily basis so that the log files don't grow too large and take up too much disk space. This process is called *checkpointing*. To do this, use the `nisping -C` command.

Security Overview



This chapter gives an overview of how NIS+ protects its namespace.

<i>Solaris Security—Overview</i>	<i>page 53</i>
<i>NIS+ Security—Overview</i>	<i>page 56</i>
<i>NIS+ Authentication and Credentials—Introduction</i>	<i>page 59</i>
<i>NIS+ Authorization and Access—Introduction</i>	<i>page 63</i>
<i>The NIS+ Administrator</i>	<i>page 68</i>
<i>NIS+ Password, Credential, and Key Commands</i>	<i>page 69</i>

Solaris Security—Overview

In essence, Solaris security is provided by gates that users must pass through in order to enter the Solaris environment, and permission matrixes that determine what they are able to do once inside. (See Figure 4-1 on page 54 for a schematic representation of this system.)

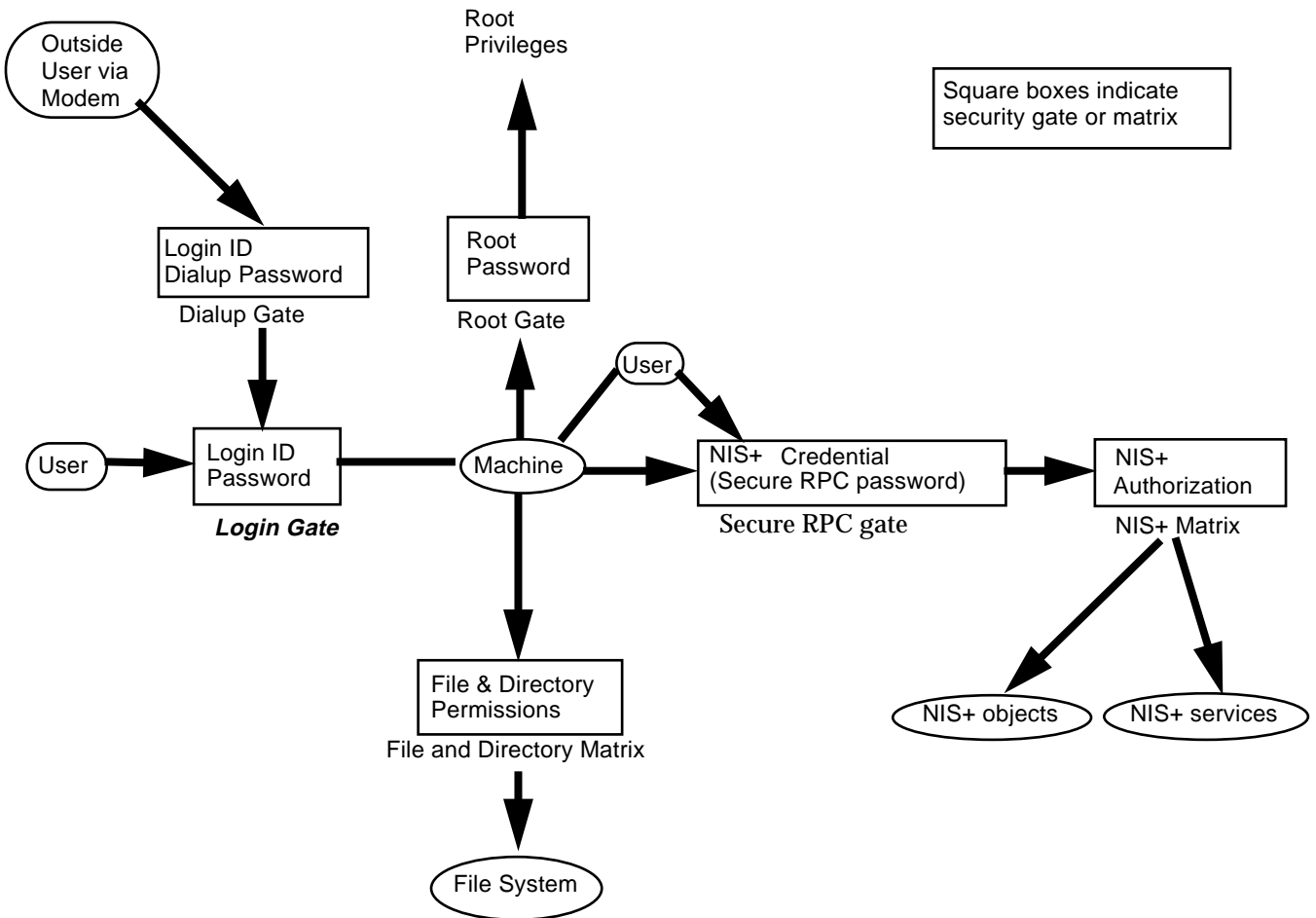


Figure 4-1 An Example Solaris Security Gates and Filters

As you can see in Figure 4-1, the overall system is composed of four gates and two permission matrixes:

- *Dialup gate.* To access a given Solaris environment from the outside through a modem and phone line, you must provide a valid Login ID and Dialup password.

Consult your Solaris documentation for detailed descriptions of the Dialup, Login, and Root gates and the File and Directory permissions matrix.

- *Login gate.* To enter a given Solaris environment you must provide a valid login ID and user password.
- *File and Directory Matrix.* Once you have gained access to a Solaris environment, your ability to read, execute, modify, create, and destroy files and directories is governed by the applicable permissions matrix.
- *Root gate.* To gain access to root privileges, you must provide a valid super user (root) password.
- *Secure RPC gate.* In an NIS+ environment running at security level 2 (the default), when you try to use NIS+ services and gain access to NIS+ objects (servers, directories, tables, table entries, and so forth.) your identity is confirmed by NIS+ using the Secure RPC process.

In some contexts **Secure RPC** passwords have been referred to as **network passwords**

To enter the Secure RPC gate requires presentation of a Secure RPC password. Your Secure RPC password and your login password normally are identical and when that is the case you are passed through the gate automatically without having to re-enter your password. (See “Secure RPC Password versus Login Password Problem” on page 82 for information regarding cases where the two passwords are not the same.)

A set of *credentials* are used to automatically pass your requests through the Secure RPC gate. The process of generating, presenting, and validating your credentials is called *authentication* because it confirms who you are and that you have a valid Secure RPC password. This authentication process is automatically performed every time you request a NIS+ service.

In Secure RPC terminology, any user without a valid credential is considered a member of the nobody class. See “Authorization Classes” on page 63 for a description of the four classes.

In an NIS+ environment running in NIS-compatibility mode (also known as YP-compatibility mode), the protection provided by the Secure RPC gate is significantly weakened because everyone has read rights for all NIS+ objects and modify rights for those entries that apply to them regardless of whether or not they have a valid credential (that is, regardless of whether or not the authentication process has confirmed their identity and validated their Secure RPC password). Since that allows *anyone* to have read rights for all NIS+ objects and modify rights for those entries that apply to them, an NIS+ network running in compatibility mode is less secure than one running in normal mode.

For details on how to create and administer NIS+ authentication and credentials, see Chapter 5, “Administering NIS+ Credentials”.

- *NIS+ objects matrix.* Once you have been properly authenticated to NIS+ your ability to read, modify, create, and destroy NIS+ objects is governed by the applicable permissions matrix. This process is called NIS+ *authorization*.

For details NIS+ permissions and authorization, see Chapter 7, “Administering NIS+ Access Rights”.

NIS+ Security—Overview

NIS+ security is an integral part of the NIS+ namespace. You cannot set up security and the namespace independently. For this reason, instructions for setting up security are woven through the steps used to set up the other components of the namespace. Once an NIS+ security environment has been set up, you can add and remove users, change permissions, reassign group members, and all other routine administrative tasks needed to manage an evolving network.

The security features of NIS+ protect the information in the namespace, as well as the structure of the namespace itself, from unauthorized access. Without these security features, any NIS+ client could obtain and change information stored in the namespace or even damage it.

NIS+ security does two things:

- *Authentication.* Authentication is used to identify NIS+ principals. Every time a principal (user or machine) tries to access an NIS+ object, the user’s identity and Secure RPC password is confirmed and validated.
- *Authorization.* Authorization is used to specify access rights. Every time NIS+ principals try to access NIS+ objects, they are placed in one of four authorization classes (owner, group, world, nobody). The NIS+ security system allows NIS+ administrators to specify different read, modify, create, or destroy rights to NIS+ objects for each class. Thus, for example, a given class could be permitted to modify a particular column in the passwd table but not read that column, or a different class could be allowed to read some entries of a table but not others.

In essence, then, NIS+ security is a two-step process:

1. *Authentication.* NIS+ uses credentials to confirm that you are who you claim to be.

(You should not have to enter a password as part of the authentication process. However, if for some reason your Secure RPC password is different than your login password, you will have to perform a `keylogin` the first time you try to access NIS+ objects or services. To perform a `keylogin`, you must provide a valid Secure RPC password. See “Secure RPC Password versus Login Password Problem” on page 82.)

2. *Authorization.* Once your identity is established by the authentication process, NIS+ determines your class. What you can do with a given NIS+ object or service depends on which class you belong to. For example, a given NIS+ table may allow one class to both read and modify the information in the table, but a different class is only allowed to read the information, and a third class is not even allowed to do that. This is similar in concept to the standard UNIX file and directory permissions system. (See “Authorization Classes” on page 63 for more information on classes.)

This process, for example, prevents someone with root privileges on machine A from using the `su` command to assume the identity of a second user who is either not logged in at all or logged in on machine B, and then accessing NIS+ objects with the second user’s NIS+ access privileges.

Note, however, that NIS+ cannot prevent someone who knows another user’s login password from assuming that other user’s identity and NIS+ access privileges. Nor can NIS+ prevent a user with root privileges from assuming the identity of another user who is logged in from the *same* machine.

Figure 4-2 details this process:

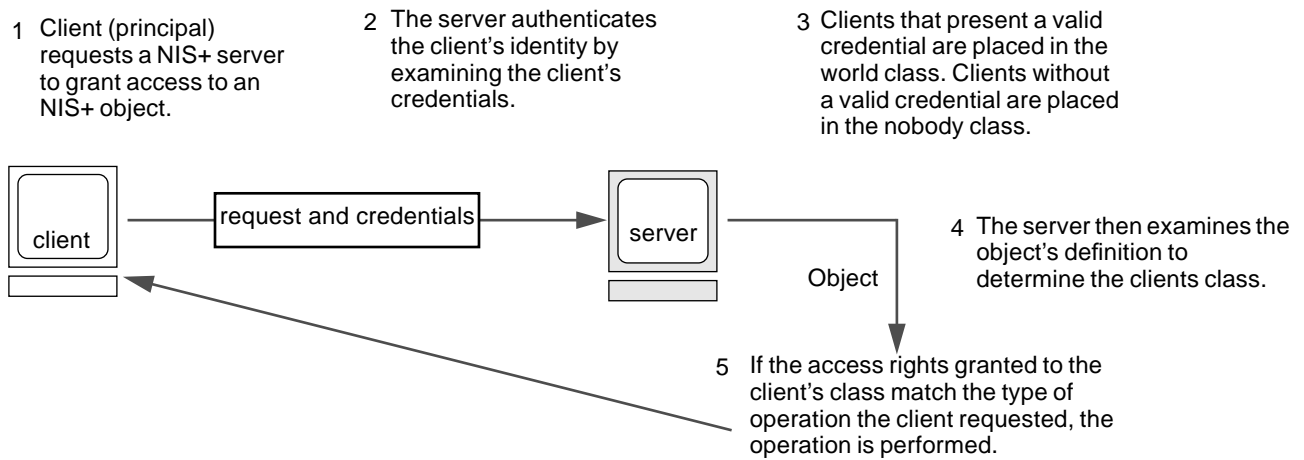


Figure 4-2 Summary of the NIS+ Security Process

NIS+ Principals

NIS+ principals are the entities (clients) that submit requests for NIS+ services. An NIS+ principal may be someone who is logged in to a client machine as a regular user, someone who is logged in as superuser, or any process that runs with superuser permission on an NIS+ client machine. Thus, an NIS+ principal can be a client user or a client workstation.

A NIS+ principal can also be the entity that supplies an NIS+ service from an NIS+ server. Since all NIS+ servers are also NIS+ clients, much of this discussion also applies to servers.

NIS+ Security Levels

NIS+ servers operate at one of two security levels. These levels determine the types of credential principals must submit for their requests to be authenticated. NIS+ is designed to run at the most secure level, which is security level 2. Level 0 is provided only for testing, setup, and debugging purposes. These security levels are summarized in Table 4-1 on page 59.

Table 4-1 NIS+ Security Levels

Security Level	Description
0	Security level 0 is designed for testing and setting up the initial NIS+ namespace. An NIS+ server running at security level 0 grants any NIS+ principal full access rights to all NIS+ objects in the domain. Level 0 is for setup purposes only and should only be used by administrators for that purpose. Level 0 should not be used on networks in normal operation by regular users.
1	Security level 1 uses AUTH_SYS security. This level is not supported by NIS+ and should not be used.
2	Security level 2 is the default. It is the highest level of security currently provided by NIS+. It authenticates only requests that use DES credentials. Requests with no credentials are assigned to the nobody class and have whatever access rights that have been granted to that class. Requests that use invalid DES credentials are retried. After repeated failure to obtain a valid DES credential, requests with invalid credentials fail with an authentication error. (A credential might be invalid for a variety of reasons such as the principal making the request is not keylogged in on that machine, the clocks are out of synch, there is a key mismatch, and so forth.)

Security Levels and Password Commands

In Solaris releases 2.0 through 2.4, you used the `nispasswd` to change your password. However, `nispasswd` could not function without credentials. (In other words, it could not function under security level 0 unless there were credentials existing from some previous higher level.) In Solaris release 2.5 (and later), the `passwd` command should now be used to change your own password regardless of security level or credential status.

NIS+ Authentication and Credentials—Introduction

The purpose of NIS+ credentials is to authenticate (confirm) the identity of each principal requesting a NIS+ service or access to a NIS+ object. In essence, the NIS+ credential/authorization process is an implementation of the Secure RPC system.

The credential/authentication system prevents someone from assuming some other user's identity. That is, it prevents someone with root privileges on one machine from using the `su` command to assume the identity of a second user who is either not logged in at all or logged in on another machine and then accessing NIS+ objects with the second user's NIS+ access privileges.



Caution – NIS+ cannot prevent someone who knows another user's login password from assuming that other user's identity and the other user's NIS+ access privileges. Nor can NIS+ prevent a user with root privileges from assuming the identity of another user who is currently logged in on the *same* machine.

Once a server authenticates a principal, it then checks the NIS+ object that the principal wants to access to see what activities that principal is authorized to perform. (See "NIS+ Authorization and Access—Introduction" on page 63 for further information on authorization.)

User and Machine Credentials

There are two basic types of principal, *users* and *machines*, and thus two different types of credentials:

- *User credentials*. When someone is logged in to an NIS+ client as a regular user, requests for NIS+ services include that person's *user* credentials.
- *Machine credentials*. When a user is logged in to an NIS+ client as superuser, request for services use the *client workstation's* credentials.

DES versus LOCAL Credentials

NIS+ principals can have two types of credential: DES and LOCAL.

DES Credentials

DES credentials are only one method of achieving authentication. In the future, other methods may be available. Thus, do not equate DES credentials with NIS+ credentials.

DES (Data Encryption Standard) credentials are the type of credential that provide secure authentication. When this guide refers to NIS+ checking a credential to authenticate a NIS+ principal, it is the DES credential that NIS+ is validating.

Each time a principal requests a NIS+ service or access to a NIS+ object, the software uses the credential information stored for that principal to generate a credential for that principal. DES credentials are generated from information created for each principal by a NIS+ administrator, as explained in Chapter 5, “Administering NIS+ Credentials.”

- When the validity of a principal’s DES credential is confirmed by NIS+, that principal is *authenticated*.
- A principal must be authenticated in order to be placed in the owner, group, or world authorization classes. In other words, you must have a valid DES credential in order to be placed in one of those classes. (Principals who do not have a valid DES credential are automatically placed in the nobody class.)
- DES credential information is always stored in the cred table of the principal’s home domain, regardless of whether that principal is a client user or a client workstation.

LOCAL Credentials

LOCAL credentials are simply a map between a user’s User ID number and NIS+ principal name which includes their home domain name. When users log in, the system looks up their LOCAL credential, which identifies their home domain where their DES credential is stored. The system uses that information to get the user’s DES credential information.

When users log in to a remote domain, those requests use their LOCAL credential which points back to their home domain; NIS+ then queries the user’s home domain for that user’s DES credential information. This allows a user to be authenticated in a remote domain even though the user’s DES credential information is not stored in that domain.

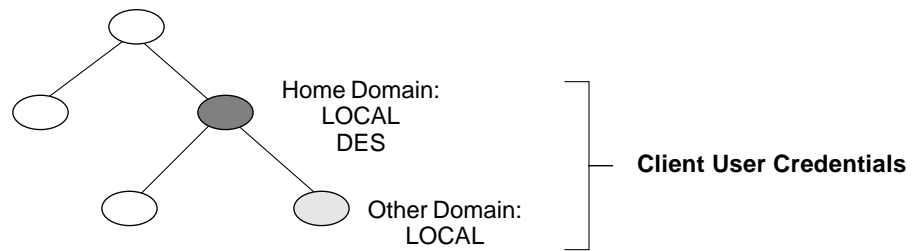


Figure 4-3 Credentials and Domains

LOCAL credential information can be stored in any domain. In fact, in order to log into a remote domain and be authenticated, a client user *must* have a LOCAL credential in the cred table of the remote domain. If a user does not have a LOCAL credential in a remote domain the user is trying to access, NIS+ will be unable to locate the user’s home domain to obtain the user’s DES credential. In such a case the user would not be authenticated and would be placed in the nobody class.

User Types and Credential Types

A user can have both types of credential, but a machine can only have a DES credential.

Root cannot have NIS+ access, as root, to other machines because the root UID of every machine is always zero. If root (UID=0) of machine A tried to access machine B as root, that would conflict with machine B’s already existing root (UID=0). Thus, a LOCAL credential doesn’t make sense for a client *workstation*; so it is allowed only for a client *user* :

Table 4-2 Types of Credentials

Type of Credential	Client User	Client Workstation
DES	Yes	Yes
LOCAL	Yes	No

NIS+ Authorization and Access—Introduction

The basic purpose of NIS+ authorization is to specify the access rights that each NIS+ principal has for each NIS+ object and service.

Once the principal making an NIS+ request is authenticated, NIS+ places them in an authorization class. The access rights (permissions) that specify which activities a principal may do with a given NIS+ object are assigned on a class basis. In other words, one authorization class may have certain access rights while a different class has different rights.

- *Authorization classes.* There are four authorization classes: owner, group, world, and nobody. (See “Authorization Classes” below for details.)
- *Access rights.* There are four types of access rights (permissions): create, destroy, modify, and read. (See “NIS+ Access Rights” on page 67 for details.)

Authorization Classes

NIS+ objects do not grant access rights directly to NIS+ principals. Instead, they grant access rights to four *classes of principal*:

- *Owner.* The principal who happens to be the object’s owner gets the rights granted to the owner class.
- *Group.* Each NIS+ object has one group associated with it. The members of an object’s group are specified by the NIS+ administrator. The principals who belong to the object’s group class get the rights granted to the group class.
- *World.* The world class encompasses all NIS+ principals that a server has been able to authenticate. (That is, everyone who has been authenticated but who is not in either the owner or group classes.)

In this context **group** refers to NIS+ groups, not UNIX or net groups. (See “The Group Class” on page 65 for a description of NIS+ groups.)

- *Nobody*. Everyone belongs to the nobody class even those who are not authenticated.

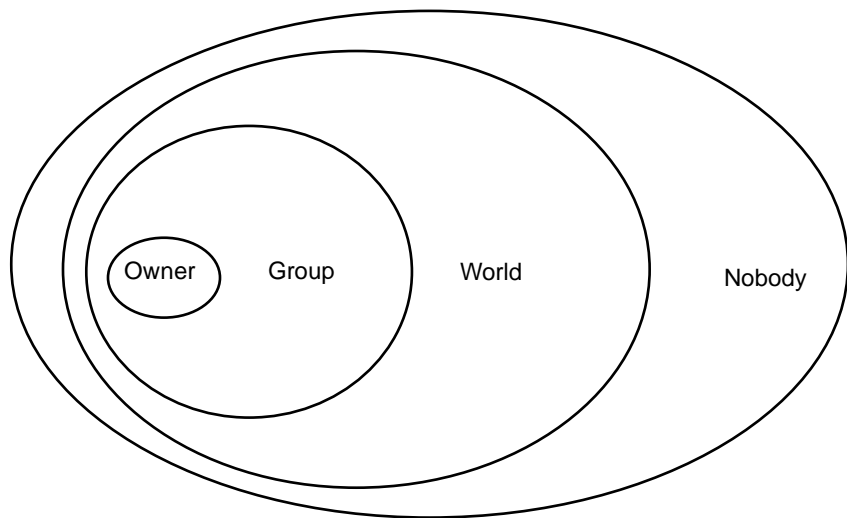


Figure 4-4 Authorization Classes

For any NIS+ request, the system determines which class the requesting principal belongs to and the principal then can use whatever access rights belonging to that class.

An object can grant any combination of access rights to each of these classes. Normally, however, a higher class is assigned the same rights as all the lower classes, plus possible additional rights.

For instance, an object could grant read access to the nobody and world classes, both read and modify access to the group class, and read, modify, create, and destroy access to the owner class.

The four classes are described in detail below.

The Owner Class

The owner is a *single* NIS+ principal.

A principal making a request for access to an NIS+ object must be authenticated (present a valid DES credential) before being granted owner access rights.

By default, an object's owner is the principal that created the object. However, an object's owner can cede ownership to another principal in two ways:

- One way is for the principal to specify a different owner at the time the object is created (see "Specifying Access Rights in Commands" on page 120).
- A second way is for the principal to change the ownership of the object after it is created (see "Changing Ownership of Objects and Entries" on page 132).

Once a principal gives up ownership, that principal gives up all owner's access rights to the object and keeps only the rights the object assigns to either the group, the world, or nobody.

The Group Class

The object's group is a *single* NIS+ group.

A principal making a request for access to an NIS+ object must be authenticated (present a valid DES credential) and belong to the group before being granted group access rights.

An NIS+ group is a collection of NIS+ principals, grouped together as a convenience for providing access to the namespace. The access rights granted to an NIS+ group apply to all the principals that are members of that group. (An object's owner, however, does not need to belong to the object's group.)

When an object is created it may be assigned a default group. A nondefault group can be specified for an object when it is created or later. An object's group may be changed at any time.

Information about NIS+ groups is stored in NIS+ group objects, under the `groups_dir` subdirectory of every NIS+ domain:

In this context **group** refers to NIS+ groups, not UNIX or net groups

Note that information about NIS+ groups is not stored in the NIS+ group table. That table stores information about UNIX groups.

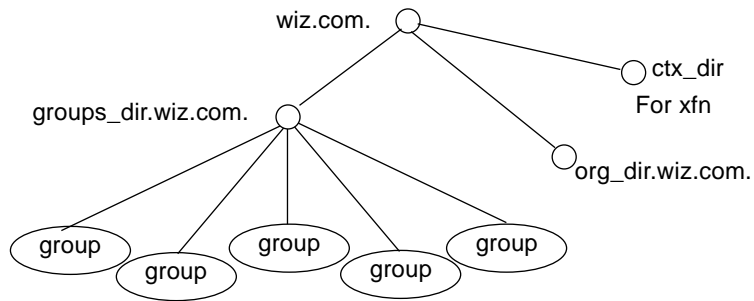


Figure 4-5 NIS+ Directory Structure

Instructions for administering NIS+ groups are provided in Chapter 9, “Administering NIS+ Groups.”

The World Class

The world class contains all NIS+ principals that are authenticated by NIS+. In other words, everyone in the owner and group class, plus everyone else who presents a valid DES credential.

Access rights granted to the world class apply to all authenticated principals.

The Nobody Class

The nobody class is composed of anyone who is not properly authenticated. In other words, anyone who does not present a valid DES credential.

Authorization Classes and the NIS+ Object Hierarchy

There is a hierarchy of NIS+ objects and authorization classes that can apply independently to each level. The standard default NIS+ directory hierarchy is:

- *Directory level.* In each NIS+ domain there are two NIS+ directory objects: `groups_dir` and `org_dir`. Each `groups_dir` directory object contains various groups. Each `org_dir` directory object contains various tables.
- *Group level or table level.* Groups contain individual entries and possibly other groups. Tables contain both columns and individual entries.

- *Column level.* A given table will have one or more columns.
- *Entry (row) level.* A given group or table will have one or more entries.

The four authorization classes apply at each level. Thus, a directory object will have its own owner and group. The individual tables within a directory object will have their own individual owners and groups which may be different than the owner and group of the directory object. Within a table, a column or an entry may have its own individual owner or group which may be different than the owner and group of the table as a whole or the directory object as a whole.

NIS+ Access Rights

NIS+ objects specify their access rights as part of their object definitions. (You can examine these by using the `niscat -o` command, described on page 172.)

NIS+ objects specify access rights for NIS+ principals in the same way that UNIX files specify permissions for UNIX users. Access rights specify the types of operations that NIS+ principals are allowed to perform on NIS+ objects.

NIS+ operations vary among different types of objects, but they all fall into one of the four access rights categories: read, modify, create, and destroy.

- *Read.* A principal with read rights to an object can view the contents of that object.
- *Modify.* A principal with modify rights to an object can change the contents of that object.
- *Destroy.* A principal with destroy rights to an object can destroy or delete the object.
- *Create.* A principal with create rights to a higher level object can create new objects within that level. In other words, if you have create rights to a NIS+ directory object, you can create new tables within that directory. If you have create rights to a NIS+ table, you can create new columns and entries within that table.

Every communication from an NIS+ client to an NIS+ server is, in effect, a request to perform one of these operations on a specific NIS+ object. For instance, when an NIS+ principal requests the IP address of another workstation, it is effectively requesting read access to the hosts table object,

which stores that type of information. When a principal asks the server to add a directory to the NIS+ namespace, it is actually requesting modify access to the directory's parent object.

Keep in mind that these rights logically evolve down from directory to table to table column and entry levels. For example, to create a new table, you must have create rights for the NIS+ directory object where the table will be stored. When you create that table, you become its default owner. As owner, you can assign yourself create rights to the table which allows you to create new entries in the table. If you create new entries in a table, you become the default owner of those entries. As table owner, you can also grant table-level create rights to others. For example, you can give your table's group class table-level create rights. In that case, any member of the table's group can create new entries in the table. The individual member of the group who creates a new table entry becomes the default owner of that entry.

The NIS+ Administrator

An NIS+ administrator is anyone who has *administrative rights* over an NIS+ object. For the purpose of this discussion, administrative rights are defined as create, destroy, and for some objects, modify rights. (See "NIS+ Access Rights" on page 67 for a description of NIS+ access rights.)

Whoever creates an NIS+ object sets the initial access rights to that object. If the creator restricts administrative rights to the object's owner (initially the creator), then only the owner has administrative power over that object. On the other hand, if the creator grants administrative rights to the object's group, then everyone in that group has administrative power over that object.

Thus, who ever has administrative rights over an object is considered to be an NIS+ administrator for that object.

In other words, the NIS+ software does not enforce any requirement that there be a single NIS+ administrator.

Theoretically, you could grant administrative rights to the world class, or even the nobody class. The software allows you to do that. But granting administrative rights beyond the group class effectively nullifies NIS+ security. Thus, if you grant administrative rights to either the World or the nobody class you are, in effect, defeating the purpose of NIS+ security.

NIS+ Password, Credential, and Key Commands

Use the following commands to administer passwords, credentials, and keys (see the appropriate man pages for a full description of each command):

- `chkey`. Changes a principal's Secure RPC key pair. Do not use `chkey` unless necessary, use `passwd` instead. See "Changing Keys for a NIS+ Principal" on page 100 for more information.
- `keylogin`. Decrypts and stores a principal's secret key with the `keyserv`. See "Keylogin" on page 99 for more information.
- `keylogout`. Deletes stored secret key from `keyserv`.
- `keyserv`. Enables the server for storing private encryption keys. See "Keylogin" on page 99 for more information.
- `newkey`. Creates a new key pair in public-key database.
- `nisaddcred`. Creates credentials for NIS+ principals. See "Creating Credential Information" on page 86 and "Administering NIS+ Credential Information" on page 96 for more information.
- `nisupdkeys`. Updates public keys in directory objects. See "Updating Public Keys" on page 105 for more information.
- `passwd`. Changes and administers principal's password. See Chapter 8, "Administering Passwords" for more information.

Part 2—Administering NIS+

This part of the manual describes how to administer an NIS+ namespace.

<i>Administering NIS+ Credentials</i>	<i>page 73</i>
<i>Administering NIS+ Keys</i>	<i>page 99</i>
<i>Administering NIS+ Access Rights</i>	<i>page 109</i>
<i>Administering Passwords</i>	<i>page 137</i>
<i>Administering NIS+ Groups</i>	<i>page 173</i>
<i>Administering NIS+ Directories</i>	<i>page 183</i>
<i>Administering NIS+ Tables</i>	<i>page 207</i>
<i>The Name Service Switch</i>	<i>page 233</i>

Administering NIS+ Credentials



Some NIS+ security tasks can be performed more easily with Solstice AdminSuite tools, if you have them available.

This chapter assumes that you have an adequate understanding of the NIS+ security system in general, and in particular of the role that credentials play in that system (see Chapter 4, “Security Overview,” for this information)

This chapter provides the following general information about credentials.

<i>How Credentials Work</i>	<i>page 74</i>
<i>Credential versus Credential Information</i>	<i>page 74</i>
<i>Authentication Components</i>	<i>page 75</i>
<i>How Principals are Authenticated</i>	<i>page 75</i>
<i>The DES Credential in Detail</i>	<i>page 79</i>
<i>Where Credential-Related Information Is Stored</i>	<i>page 84</i>
<i>The Cred Table in Detail</i>	<i>page 85</i>

This chapter then describes how to use the NIS+ credential administration commands to perform the following tasks.

<i>Creating Credential Information</i>	<i>page 86</i>
<i>The nisaddcred Command</i>	<i>page 87</i>
<i>How nisaddcred Creates Credential Information</i>	<i>page 89</i>
<i>The Secure RPC Netname and NIS+ Principal Name</i>	<i>page 90</i>
<i>Creating Credential Information for the Administrator</i>	<i>page 91</i>

<i>Creating Credential Information for NIS+ Principals</i>	<i>page 91</i>
<i>Updating Your Own Credential Information</i>	<i>page 97</i>
<i>Removing Credential Information</i>	<i>page 97</i>

For a complete description of the commands used to perform these tasks their syntax and options, see the NIS+ man pages.

How Credentials Work

This section describes how the credential and authentication process works.

The credential/authentication system prevents someone from assuming some other user's identity. That is, it prevents someone with root privileges on one machine from using the `su` command to assume the identity of a second user who is either not logged in at all or logged in on another machine and then accessing NIS+ objects with the second user's NIS+ access privileges.



Caution – NIS+ cannot prevent someone who knows another user's login password from assuming that other user's identity and the other user's NIS+ access privileges. Nor can NIS+ prevent a user with root privileges from assuming the identity of another user who is currently logged in on the *same* machine.

See Chapter 4, "Security Overview," for a description of how NIS+ credentials and authentication work with authorization and access rights to provide security for the NIS+ namespace.

Credential versus Credential Information

To understand how DES credentials are created and how they work, you need to distinguish between the credential itself and the information that is used to create and verify it.

- *Credential information*: The data that is used to generate a DES credential and by the server to verify that credential.

- *DES credential*: The bundle of numbers that is sent by the principal to the server to authenticate the principal. A principal's credential is generated and verified each time the principal makes an NIS+ request. See "The DES Credential in Detail" on page 79 for a detailed description of the DES credential.

Authentication Components

In order for the credential/authentication process to work the following components must be in place:

- *Principal's DES credential information*. This information is initially created by an NIS+ administrator for each principal. It is stored in the cred table of the principal's home domain. A principal's DES credential information consists of:
 - *Principal name*. This would be a user's fully qualified login ID or a machine's fully qualified host name.
 - *Principal's Secure RPC netname*. Each principal has a unique Secure RPC netname. (See "DES Credential Secure RPC Netname" on page 80 for more information on Secure RPC netnames.)
 - *Principal's public key*.
 - *Principal's encrypted private key*.
- *Principal's LOCAL credential*.
- *Server's public keys*. Each directory object stores copies of the public keys of all the servers in that domain. Note that each server's DES credentials are also stored in the cred table.
- *Keyserver copy of principal's private key*. The keyserver has a copy of the private key of the principal that is currently logged in (user or machine).

How Principals are Authenticated

There are three phases to the authorization process:

- *Preparation phase*. This consists of the setup work performed by an NIS+ administrator prior to the user logging in; for example, creating credential information for the user.
- *Login phase*. This consists of the actions taken by the system when a user logs in.

- *Request phase.* This consists of the actions taken by the software when a NIS+ principal makes a request for a NIS+ service or access to a NIS+ object.

These three phases are described in detail in the following subsections.

Credentials Preparation Phase

The easiest way for an NIS+ administrator to create credential information for users is to use the `nisclient` script as described in the `nisclient` man page.

Prior to an NIS+ principal logging in, an NIS+ administrator must create DES credential information for that principal (user or machine). The administrator must:

- Create a public key and an encrypted private key for each principal. These keys are stored in the principal's home domain cred table. This can be done with the `nisaddcred` command as described in "Creating Credential Information for NIS+ Principals" on page 91.
- Create server public keys. (See "Updating Public Keys" on page 105.)

Login Phase—Detailed Description

When a principal logs into the system the following steps are automatically performed:

1. The `keylogin` program is run for the principal. The `keylogin` program gets the principal's encrypted private key from the cred table and decrypts it using the principal's login password.

Note – When a principal's login password is different from his or her Secure RPC password, `keylogin` cannot decrypt it and the user starts getting "cannot decrypt" errors or the command fails without a message. For a discussion of this problem, see "Secure RPC Password versus Login Password Problem" on page 82.

2. The principal's decrypted private key is passed to the keyserver which stores it for use during the request phase.

Note – The decrypted private key remains stored for use by the keyserver until the user does an explicit `keylogout`. If the user simply logs out (or goes home for the day without logging out), the decrypted private key remains stored in the server. If someone with root privileges on a user's machine switched to the user's login ID, that person would then have use of the user's decrypted

private key and could access NIS+ objects using the user's access authorization. Thus, for added security, users should be cautioned to perform an *explicit* `keylogout` when they cease work. If they also log out of the system, all they need do is log back in when they return. If they do not explicitly log out, they will have to perform an explicit `keylogin` when they return to work.

Request Phase—Detailed Description

Every time a NIS+ principal requests access to an NIS+ object, the NIS+ software performs a multistep process to authenticate that principal:

1. NIS+ checks the cred table of the object's domain. If:
 - The principal has LOCAL credential information, NIS+ uses the domain information contained in the LOCAL credential to find the principal's home domain cred table where it obtains the information used in Step 7.
 - The principal has no credential information, the rest of the process is aborted and the principal is given the authorization access class of nobody.
2. NIS+ gets the user's DES credential from the cred table of the user's home domain. The encrypted private key is decrypted with the user's password and saved by the keyserver.
3. NIS+ obtains the server's public key from the NIS+ directory object.
4. The keyserver takes the principal's decrypted private key and the public key of the object's server (the server where the object is stored) and uses them to create a *common key*.
5. The common key is then used to generate an encrypted *DES key*. To do this, Secure RPC generates a random number which is then encrypted using the common key. For this reason, the DES key is sometimes referred to as the *random key* or the *random DES key*.
6. NIS+ then takes the current time of the principal's server and creates a time stamp that is encrypted using the DES key.
7. NIS+ then creates a 15-second window, which is encrypted with the DES key. This window is the maximum amount of time that is permitted between the time stamp and the server's internal clock.

8. NIS+ then forms the principal's DES credential, which is composed of the following:
 - The principal's Secure RPC netname (`unix.identifier@domain`) from the principal's cred table (see "DES Credential Secure RPC Netname" on page 80 for more detail on the netname).
 - The principal's encrypted DES key from the keyserver
 - The encrypted time stamp
 - The encrypted window
9. NIS+ then passes the following information to the server where the NIS+ object is stored:
 - The access request (whatever it might be)
 - The principal's DES credential
 - Window verifier (encrypted), which is the encrypted window plus one
10. The object's server receives this information.
11. The object's server uses the Secure RPC netname portion of the credential to look up the principal's public key in the cred table of the principal's home domain.
12. The server then uses the principal's public key and the server's private key to regenerate the common key. This common key must match the common key that was generated by the principal's private key and the server's public key.
13. The common key is used to decrypt the DES key that arrived as part of the principal's credential.
14. The server decrypts the principal's time stamp with the newly decrypted DES key and verifies it with the window verifier.
15. The server then compares the decrypted and verified time stamp with the server's current time and proceeds as follows:
 - a. If the time difference at the server *exceeds* the window limit, the request is denied and the process aborts with an error message. For example, suppose the time stamp is 9:00am and the window is one minute. If the request is received and decrypted by the server after 9:01am, it is denied.

-
- b. If the time stamp is within the window limit, the server checks to see if the time stamp is *greater* than the one previously received from the principal. This ensures that NIS+ requests are handled in the correct order.
 - i. Requests received out of order are rejected with an error message. For example, if the time stamp is 9:00am and the most recently received request from this principal had a time stamp of 9:02am, the request would be rejected.
 - ii. Requests that have a time stamp equal to the previous one are rejected with an error message. This ensures that a replayed request is not acted on twice. For example, if the time stamp is 9:00am and the most recently received request from this principal also had a time stamp of 9:00am, this request would be rejected.
 - c. If the time stamp is within the window limit, and greater than the previous request from that principal, the server accepts the request.
16. The server then complies with the request and stores the time stamp from this principal as the most recently received and acted on request.
 17. To confirm to the principal that the information received from the server in answer to the request comes from a trusted server, the server encrypts the time stamp with the principal's DES key and sends it back to the principal along with the data.
 18. At the principal's end, the returned time stamp is decrypted with the principal's DES key.
 - If the decryption succeeds, the information from the server is returned to the requester.
 - If the decryption fails for some reason, an error message is displayed.

The DES Credential in Detail

The DES credential consists of:

- The principal's Secure RPC netname (see "DES Credential Secure RPC Netname" below).
- A verification field (see "DES Credential Verification Field," below).

DES Credential Secure RPC Netname

Remember that an NIS+ principal name **always** has a trailing dot, while a Secure RPC netname **never** does.

- **Secure RPC netname.** This portion of the credential is used to identify the NIS+ principal. Every Secure RPC netname contains three components:
 - **Prefix.** The prefix is always the word `unix`.
 - **Identifier.** If the principal is a client user, the ID field is the user's UID. If the principal is a client workstation, the ID field is the workstation's hostname.
 - **Domain name.** The domain name is the name of the domain that contains the principal's DES credential (in other words, the principal's home domain).

Table 5-1 Secure RPC Netname Format

Principal	Prefix	Identifier	Domain	Example
User	unix	UID	Domain containing user's password entry and the DES credential itself	unix.24601@sales.wiz.com
Workstation	unix	hostname	The domain name returned by executing the <code>domainname</code> command on that workstation	unix.machine7@sales.wiz.com

DES Credential Verification Field

The verification field is used to make sure the credential is not forged. It is generated from the credential information stored in the cred table.

The verification field is composed of:

- The principal's encrypted DES key, generated from the principal's private key and the NIS+ server's public key as described on page 77
- The encrypted time stamp
- The time window

How the DES Credential Is Generated

See Figure 5-2 on page 82.

To generate its DES credential, the principal depends on the `keylogin` command, which must have been executed *before* the principal tries to generate its credential. The `keylogin` command (often referred to simply as a *keylogin*) is executed automatically when an NIS+ principal logs in.

Note – Note that if the principal’s login password is different from the principal’s Secure RPC password, a successful `keylogin` cannot be performed. See “Secure RPC Password versus Login Password Problem” on page 82 for a discussion of this situation.

The purpose of the `keylogin` is to give the principal access to the principal’s private key. `keylogin` fetches the principal’s private key from the cred table, decrypts it with the principal’s Secure RPC password (remember that the private key was originally encrypted with the principal’s Secure RPC password), and stores it locally with the keyserver for future NIS+ requests.

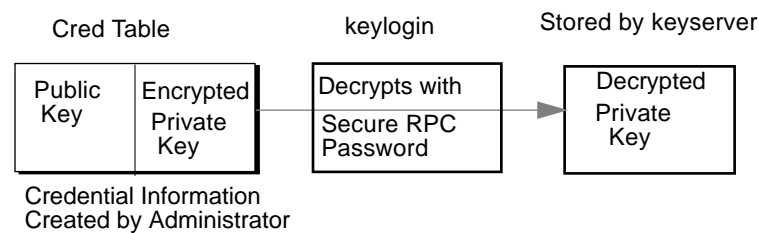


Figure 5-1 `keylogin` Generates a Principal’s Private Key

To generate its DES credential, the principal still needs the public key of the server to which it will send the request. This information is stored in the principal’s directory object. Once the principal has this information, it can form the verification field of the credential.

First, the principal generates a random DES key for encrypting various credential information. The principal uses its own private key (stored in the keyserver) and the server’s public key to generate a common key that is used to generate and encrypt the random DES key. It then generates a time stamp that is encrypted with the DES key and combines it with other credential-related information into the verification field:

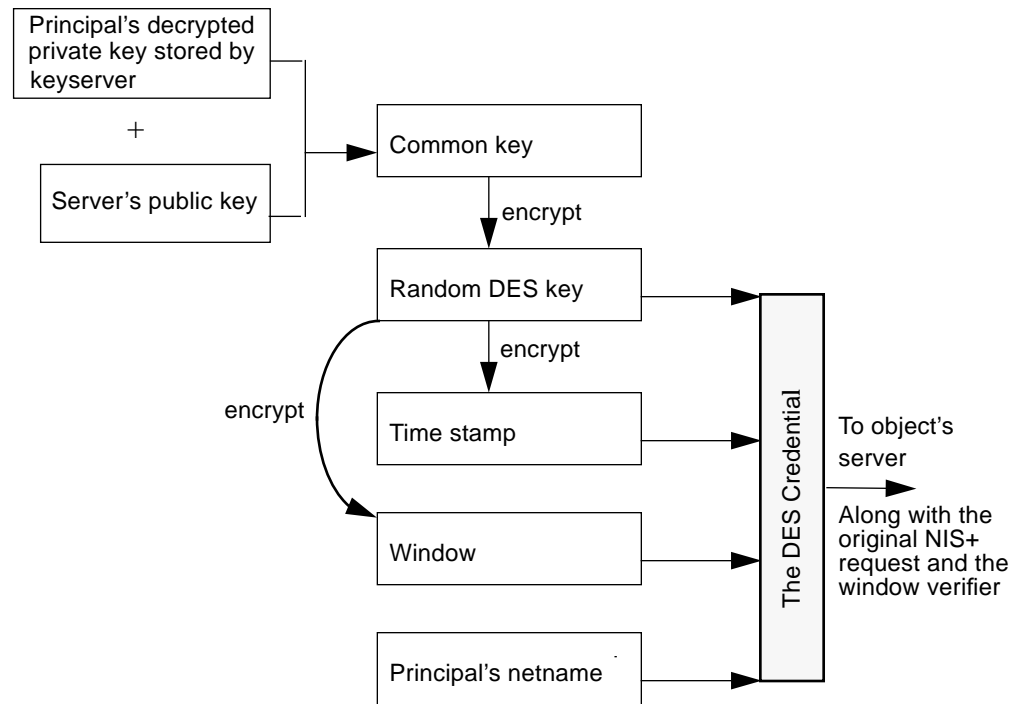


Figure 5-2 Creating the DES Credential

Secure RPC Password versus Login Password Problem

When a principal's login password is different from his or her Secure RPC password, `keylogin` cannot decrypt it at login time because `keylogin` defaults to using the principal's login password, and the private key was encrypted using the principal's Secure RPC password.

When this occurs, the principal can log in to the system, but for NIS+ purposes the principal is placed in the authorization class of nobody because the keyserver does not have a decrypted private key for that user. Since most NIS+ environments are set up to deny the nobody class create, destroy, and modify rights to most NIS+ objects, this results in "permission denied" errors when the user tries to access NIS+ objects.

In this context, **network password** is sometimes used as a synonym for Secure RPC password. When prompted for your network password, type your Secure RPC password.

To be placed in one of the other authorization classes, a user in this situation must explicitly run the `keylogin` program and give the principal's Secure RPC password when `keylogin` prompts for a password. (See "Keylogin" on page 99.)

But an explicit `keylogin` provides only a temporary solution that is good only for the current login session. The keyserver now has a decrypted private key for the user, but the private key in the user's cred table is still encrypted using the user's Secure RPC password, which is different than the user's login password. The next time the user logs in, the same problem recurs. To permanently solve the problem the user needs to re-encrypt the private key in the cred table to one based on the user's login ID rather than the user's Secure RPC password. To do this, the user needs to run `chkey -p` as described in "Changing Keys for a NIS+ Principal" on page 100.

Thus, to permanently solve problems related to a difference in Secure RPC password and login password, the user (or an administrator acting for the user) must perform these steps:

1. Login using the login password.
2. Run the `keylogin` program to temporarily get a decrypted private key stored in the keyserver and thus gain temporary NIS+ access privileges.
3. Run `chkey -p` to permanently change the encrypted private key in the cred table to one based on the user's login password.
4. When you are ready to finish this login session, run `keylogout`.
5. Log off the system with `logout`.

Cached Public Keys Problems

Occasionally, you may find that even though you have created the proper credentials and assigned the proper access rights, some principal requests still get denied. The most common cause of this problem is the existence of stale objects with old versions of a server's public key. You can usually correct this problem by:

- Running `nisupdkeys` on the domain you are trying to access. (See "The `nisupdkeys` Command" on page 105 for information on using the `nisupdkeys` command and "Stale and Outdated Credential Information" on page 322 for information on how to correct this type of problem.)

- Killing the `nis_cachmgr` on your machine, removing `/var/nis/NIS_SHARED_DIRCACHE`, and then restarting `nis_cachmgr`.

Where Credential-Related Information Is Stored

This section describes where credential-related information is stored throughout the NIS+ namespace.

Credential-related information, such as public keys, is stored in many locations throughout the namespace. NIS+ updates this information periodically, depending on the time-to-live values of the objects that store it, but sometimes, between updates, it gets out of sync. As a result, you may find that operations that should work, do not. Table 5-2 lists all the objects, tables, and files that store credential-related information and how to reset it.

Table 5-2 Where Credential-Related Information Is Stored

Item	Stores	To Reset or Change
cred table	NIS+ principal's public key and private key. These are the master copies of these keys.	Use <code>nisaddcred</code> to create new credentials; it updates existing credentials. An alternative is <code>chkey</code> .
directory object	A copy of the public key of each server that supports it.	Run the <code>/usr/lib/nis/nisupdkeys</code> command on the directory object.
keyserver	The secret key of the NIS+ principal that is currently logged in.	Run <code>keylogin</code> for a principal user or <code>keylogin -r</code> for a principal workstation.
NIS+ daemon	Copies of directory objects, which in turn contain copies of their servers' public keys.	Kill the <code>rpc.nisd</code> daemon and the cache manager and remove <code>NIS_SHARED_DIRCACHE</code> from <code>/var/nis</code> . Then restart both.
Directory cache	A copy of directory objects, which in turn contain copies of their servers' public keys.	Kill the NIS+ cache manager and restart it with the <code>nis_cachmgr -i</code> command. The <code>-i</code> option resets the directory cache from the cold-start file and restarts the cache manager.

Table 5-2 Where Credential-Related Information Is Stored (Continued)

Item	Stores	To Reset or Change
cold-start file	A copy of a directory object, which in turn contains copies of its servers' public keys.	On the root master, kill the NIS+ daemon and restart it. The daemon reloads new information into the existing NIS_COLD_START file. On a client workstation, first remove the NIS_COLD_START and NIS_SHARED_DIRCACHE files from /var/nis, and kill the cache manager. Then re-initialize the principal with <code>nisinit -c</code> . The principal's trusted server reloads new information into the workstation's NIS_COLD_START file.
passwd table	A user's password.	Use the <code>passwd -r nisplus</code> command. It changes the password in the NIS+ passwd table and updates it in the cred table.
passwd file	A user's password or a workstation's superuser password.	Use the <code>passwd -r nisplus</code> command, whether logged in as super user or as yourself, whichever is appropriate.
passwd map (NIS)	A user's password	Use the <code>passwd -r nisplus</code> command.

The Cred Table in Detail

Credential information for principals is stored in a *cred table*. The cred table is one of the 16 standard NIS+ tables. Each domain has one cred table, which stores the credential information of client workstations that belong to that domain and client users who are allowed to log into them. (In other words, the principals of that domain.) The cred tables are located in their domains' `org_dir` subdirectory.



Caution – Never link a cred table. Each `org_dir` directory should have its own cred table. Do not use a link to some other `org_dir` cred table.

For users, the cred table stores LOCAL credential information for all users who are allowed to log into any of the machines in the domain. The cred table also stores DES credential information for those users that have the domain as their home domain.

You can view the contents of a cred table with the `niscat` command, described in Chapter 11, “Administering NIS+ Tables.”

The cred table as shown in Table 5-3 has five columns:

Table 5-3 Cred Table Credential Information

	NIS+ Principal Name	Authentication Type	Authentication Name	Public Data	Private Data
Column Name	<code>cname</code>	<code>auth_type</code>	<code>auth_name</code>	<code>public_data</code>	<code>private_data</code>
User	Fully qualified principal name	LOCAL	UID	GID list	
Machine	Fully qualified principal name	DES	Secure RPC netname	Public key	Encrypted Private key

The Authentication Type column, determines the types of values found in the other four columns.

- *LOCAL*. If the authentication type is LOCAL, the other columns contain a principal user’s name, UID, and GID; the last column is empty.
- *DES*. If the authentication type is DES, the other columns contain a principal’s name, Secure RPC netname, public key, and encrypted private key. These keys are used in conjunction with other information to encrypt and decrypt a DES credential.

Creating Credential Information

There are several methods of creating and administering credential information:

- Use Solstice AdminSuite tools if you have them available. They provide easier methods of credential administration and are recommended for administering individual credentials.

- Use the `nisclient` script. This is another easy method of creating or altering credentials for a single principal. Because of its convenience, this is a recommended method of administering individual credentials. Part 1 of *NIS+ and DNS Setup and Configuration Guide* gives step by step instructions on using the `nisclient` script to create credential information.
- Use the `nispopulate` script. This is an easy method of creating or altering credentials for a one or more principals who already have information on them stored in NIS maps or `/etc` files. Because of its convenience, this is a recommended method of administering credentials for groups of NIS+ principals. Part 1 of *NIS+ and DNS Setup and Configuration Guide* gives step by step instructions on using the `nispopulate` script to create credential information.
- Use the `nisaddcred` command. The section below describes how credentials and credential information are created using `nisaddcred`.

The `nisaddcred` Command

The command used to create credential information is `nisaddcred`.

Note – You can also use the `nispopulate` and `nisclient` scripts to create credential information. They, in turn, use the `nisaddcred` command. These scripts are much easier to use, and more efficient, than the `nisaddcred` command. Unless your network requires special features, you should use the scripts.

The `nisaddcred` command creates, updates, and removes LOCAL and DES credential information. To create credential information, you must have create rights to the proper domain's cred table. To update a credential, you must have modify rights to the cred table or, at least, to that particular entry in the cred table. To delete a credential, you must have destroy rights to the cred table or the entry in the cred table.

- To create or update credentials for another NIS+ principal, use:

For LOCAL credentials

```
nisaddcred -p uid -P principal-name local
```

For DES credentials

```
nisaddcred -p rpc-netname -P principal-name des
```

- To update your own credentials, use:

For LOCAL credentials

```
nisaddcred local
```

For DES credentials

```
nisaddcred des
```

- To remove credentials, use:

```
nisaddcred -r principal-name
```

Related Commands

In addition to the `nisaddcred` command described in this chapter, two other commands can provide some useful information about credentials:

Table 5-4 Additional Credential-Related Commands

Command	Description	See
<code>niscat -o</code>	Lists a directory's properties. By looking in the public key field of the directory's server, you can tell whether the directory object is storing a public key.	page 184
<code>nismatch principal cred.org_dir</code>	When run on the cred table, displays credential information for <i>principal</i> .	page 219

How `nisaddcred` Creates Credential Information

LOCAL Credential Information

When used to create LOCAL credential information, `nisaddcred` simply extracts the principal user's UID (and GID) from the principal's login record and places it in the domain's cred table.

DES Credential Information

When used to create DES credential information, `nisaddcred` goes through a two-part process:

1. Forming the principal's Secure RPC netname. A Secure RPC netname is formed by taking the principal's user ID number from the password record and combining it with the domain name (`unix.1050@wiz.com`, for example).
2. Generating the principal's private and public keys.

To encrypt the private key, `nisaddcred` needs the principal's Secure RPC password. When the `nisaddcred` command is invoked with the `des` argument, it prompts the principal for a Secure RPC password. Normally, this password is the same as the principal's login password. (If it is different, the user will have to perform additional steps when logging in, as described in "Secure RPC Password versus Login Password Problem" on page 82.)

The `nisaddcred` command generates a pair of random, but mathematically related 192-bit authentication keys using the Diffie-Hellman cryptography scheme. These keys are called the Diffie-Hellman key-pair, or simply *key-pair* for short.

One of these is the *private key*, and the other is the *public key*. The public key is placed in the public data field of the cred table. The private key is placed in the private data field, but only after being encrypted with the principal's Secure RPC password:

nisaddcred:

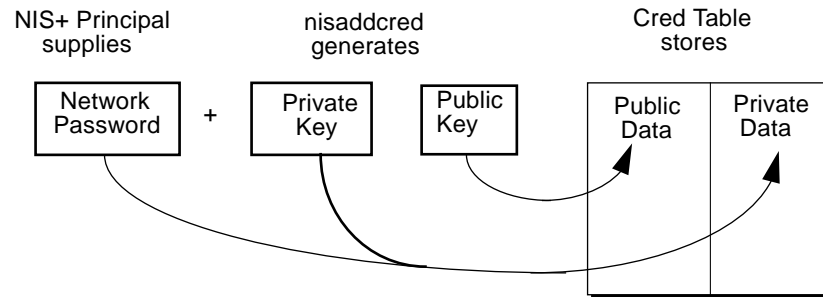


Figure 5-3 How nisaddcred Creates a Principal's Keys

The principal's private key is encrypted as a security precaution because the cred table, by default, is readable by all NIS+ principals, even unauthenticated ones.

The Secure RPC Netname and NIS+ Principal Name

When creating credential information, you will often have to enter a principal's *rpc-netname* and *principal-name*. Each has its own syntax:

- **Secure RPC netname.** A Secure RPC netname is a name whose syntax is determined by the Secure RPC protocol. Therefore, it does not follow NIS+ naming conventions:
 - For users, the syntax is: `unix.uid@domain`
 - For machines, the syntax is: `unix.hostname@domain`

If a Secure RPC netname identifies a user, it requires the user's UID. If it identifies a workstation, it requires the workstation's host name. (When used with the `nisaddcred` command it is always preceded by the `-p` (lowercase) flag.)

A Secure RPC netname always begins with the `unix` (all lowercase) prefix and ends with a domain name. However, because it follows the Secure RPC protocol, the domain name does not contain a trailing dot.

- *Principal name.* An NIS+ principal follows the normal NIS+ naming conventions, but it must always be fully qualified. the syntax is: *principal.domain*.

Whether it identifies a client user or a client workstation, it begins with the principal's *name*, followed by a dot and the complete domain name, ending in a dot. (When used with `nisaddcred` to create credential information, it is always preceded by the `-P` (uppercase) flag. When used to remove credential information, it does not use the `-P` flag.)

Creating Credential Information for the Administrator

When a namespace is first set up, credential information is created first for the administrators who will support the domain. Once they have credential information, they can create credential information for other administrators, client workstations, and client users.

When you try to create your own credential information, you run into a problem of circularity: you cannot create your own credential information unless you have Create rights to your domain's cred table, but if the NIS+ environment is properly set up, you cannot have such rights until you have credentials. You have to step out of the loop somehow. You can do this in one of two ways:

- By creating your credential information while logged in as superuser to your domain's master server
- By having another administrator create your credential information using a dummy password, then changing your password with the `chkey` command.

In either case, your credential information is thus created by another NIS+ principal. To create your own credential information, follow the instructions in "Creating Credential Information for NIS+ Principals" on page 91.

Creating Credential Information for NIS+ Principals

Credential information for NIS+ principals can be created any time after their domain has been set up; in other words, once a cred table exists.

To create credential information for an NIS+ principal:

- You must have Create rights to the cred table of the principal's home domain.
- The principal must be recognized by the server. This means that:
 - If the principal is a user, the principal must have an entry either in the domain's NIS+ passwd table or in the server's `/etc/passwd` file.
 - If the principal is a workstation, it must have an entry either in the domain's NIS+ Hosts table or in the server's `/etc/hosts` file.

Once those conditions are met, you can use the `nisaddcred` command with both the `-p` and `-P` options:

For LOCAL credentials

```
nisaddcred -p uid -P principal-name local
```

For DES credentials

```
nisaddcred -p rpc.netname -P principal-name des
```

Remember these principles:

- You can create both LOCAL and DES credential information for a principal user.
- You can only create DES credential information for a principal workstation.
- You can create DES credential information only in the principal's home domain (user or machine).
- You can create LOCAL credential information for a user in both the user's home domain and in other domains.

For User Principals—Example

This example creates both LOCAL and DES credential information for an NIS+ user named `morena` who has a UID of 11177. She belongs to the `sales.wiz.com` domain, so this example enters her credential information from a principal machine of that domain:

```
salesclient# nisaddcred -p 11177 -P morena.sales.wiz.com. local
salesclient# nisaddcred -p unix.11177@sales.wiz.com -P morena.sales.wiz.com. des
Adding key pair for unix.11177@sales.wiz.com (morena.sales.wiz.com.).
Enter login password:
```

The proper response to the `Enter login password:` prompt is `morena`'s login password. (If you don't know her login password, you can use a dummy password that she can later change using `chkey`, as described in the next example.)

Using a Dummy Password and chkey—Example

If you don't know the user's login password, you can use a dummy password as described below.

Table 5-5 on page 94 shows how another administrator, whose credential information you create using a dummy password, can then use `chkey` to change his or her own password. In this example, you create credential information for an administrator named `eiji` who has a UID of 119. `eiji` belongs to the root domain, so you would enter his credential information from the root master server which is named `rmaster`.

Table 5-5 Creating Administrator Credentials: Command Summary

Tasks	Commands
Create LOCAL credential information for eiji.	<code>rmaster# nisaddcred -p 119 -P eiji.wiz.com. local</code>
Create DES credential information for eiji.	<code>rmaster# nisaddcred -p unix.119@wiz.com -P eiji.wiz.com. des</code> Adding key pair for unix.119@wiz.com (eiji.wiz.com.).
Type dummy password for eiji. Re-enter dummy password.	Enter eiji's login password: nisaddcred: WARNING: password differs from login passwd. Retype password:
You tell eiji the dummy password that you used.	
eiji logs into rmaster. eiji enters real login password.	rmaster login: eiji Password:
eiji gets error message but is allowed to log in anyway.	Password does not decrypt secret key for unix.119@wiz.com.
eiji runs keylogin. eiji types dummy password.	rmaster% keylogin Password: <i>dummy-password</i>
eiji runs chkey -p.	rmaster% chkey -p Updating nisplus publickey database Generating new key for 'unix.119@wiz.com'.
eiji types real login password. eiji re-types real login password.	Enter login password: Retype password: Done.

First, you would create eiji's credential information in the usual way, but using a dummy login password. NIS+ would warn you and ask you to re-type it. When you did, the operation would be complete. The domain's cred table would contain eiji's credential information based on the dummy password. The domain's passwd table (or /etc/passwd file), however, would still have his login password entry so that he can log on to the system.

Then, eiji would log in to the domain's master server, typing his *correct* login password (since the login procedure checks the password entry in the passwd table or /etc/passwd file). From there, eiji would first run `keylogin`, using the dummy password (since a `keylogin` checks the cred table), and then use the `chkey -p` command to change the cred entry to the real thing.

Creating in Another Domain—Example

The two previous examples created credential information for a principal user while the principal user was logged in to the master server of the principal's home domain. However, if you have the proper access rights, you can create credential information in another domain. Simply append the domain name to this syntax:

For LOCAL credentials

```
nisaddcred -p uid -P principal-name local domain-name
```

For DES credentials

```
nisaddcred -p rpc-netname -P principal-name des domain-name
```

The following example first creates LOCAL and DES credential information for an administrator named `chou` in her home domain, which happens to be the root domain, then adds her LOCAL credential information to the `sales.wiz.com` domain. `chou`'s UID is 11155. This command is typed on from the root master server. For simplicity, it assumes you are entering `chou`'s correct login password.

```
rmaster# nisaddcred -p 11155 -P chou.wiz.com. local

rmaster# nisaddcred -p unix.11155@wiz.com -P chou.wiz.com. des
Adding key pair for unix.11155@wiz.com (chou.wiz.com.).
Enter login password:

rmaster# nisaddcred -p 11155 -P chou.wiz.com. local sales.wiz.com.
```

LOCAL credential information maps a UID to an NIS+ principal name. Although an NIS+ principal that is a client user can have different user IDs in different domains, it can have only one NIS+ principal name. So, if an NIS+ principal such as `chou` will be logging in from a domain other than her home domain, not only should she have a password entry in that domain, but also a LOCAL credential in that domain's cred table.

For Workstations—Example

This example creates credential information for a principal *workstation*. Its host name is `starshine1` and it belongs to the root domain. Therefore, its credential information is created from the root master server. In this example, you create them while logged in as root to the root master; however, if you already have valid credential information and the proper access rights, you could create them while logged in as yourself.

```
rmaster# nisaddcred -p unix.starshine1@wiz.com -P starshine1.wiz.com. des
Adding key pair for unix.starshine1@wiz.com
      (starshine1.wiz.com.).
Enter starshine1.wiz.com.'s root login password:
Retype password:
```

The proper response to the password prompt is the principal workstation's superuser password. Of course, you could use a dummy password that would later be changed by someone logged in as superuser to that principal workstation.

Administering NIS+ Credential Information

The following sections describe how to use the `nisaddcred` command to administer existing credential information. You must have create, modify, read, and destroy rights to the cred table to perform these operations.

Updating Your Own Credential Information

Updating your own credential information is considerably easier than creating it. Just type the simple versions of the `nisaddcred` command while logged in as yourself:

```
# nisaddcred des
# nisaddcred local
```

To update credential information for someone else, you simply perform the same procedure that you would use to create that person's credential information.

Removing Credential Information

The `nisaddcred` command removes a principal's credential information, but only from the local domain where the command is run.

Thus, to completely remove a principal from the entire system, you must explicitly remove that principal's credential information from the principal's home domain and all domains where the principal has LOCAL credential information.

To remove credential information, you must have modify rights to the local domain's cred table. Use the `-r` option and specify the principal with a full NIS+ principal name:

```
# nisaddcred -r principal-name
```

The following two examples remove the LOCAL and DES credential information of the administrator `morena.wiz.com`. The first example removes both types of credential information from her home domain (`wiz.com`), the second removes her LOCAL credential information from the `sales.wiz.com` domain. Note how they are each entered from the appropriate domain's master servers.

```
rmaster# nisaddcred -r morena.wiz.com.
salesmaster# nisaddcred -r morena.wiz.com.
```

To verify that the credential information was indeed removed, run `nismatch` on the `cred` table, as shown below. For more information about `nismatch`, see Chapter 11, “Administering NIS+ Tables.”

```
rmaster# nismatch morena.wiz.com. cred.org_dir
salesmaster# nismatch morena.wiz.com. cred.org_dir
```

Administering NIS+ Keys

This chapter describes how to use the `keylogin`, `chkey`, and `nisupdkeys` commands to administer keys. (The `nisaddcred` command also performs some key-related operations. See “The `nisaddcred` Command” on page 87 for more information.)

Some NIS+ security tasks can be performed more easily with Solstice AdminSuite tools if you have them available.

This chapter assumes that you have an adequate understanding of the NIS+ security system in general, and in particular of the role that keys play in that system (see Chapter 4, “Security Overview,” for this information).

<i>Keylogin</i>	<i>page 99</i>
<i>Changing Keys for a NIS+ Principal</i>	<i>page 100</i>
<i>Updating Public Keys</i>	<i>page 105</i>
<i>The <code>nisupdkeys</code> Command</i>	<i>page 105</i>
<i>Updating Public Keys Arguments and Examples</i>	<i>page 107</i>
<i>Updating IP Addresses</i>	<i>page 107</i>

Keylogin

When a principal logs in, the login process prompts for a password. That password is used to pass the user through the login security gate and give the user access to the network. The login process also decrypts the user’s private key stored in the user’s home domain cred table and passes that private key to the keyserver. The keyserver then uses that decrypted private key to authenticate the user each time the user accesses an NIS+ object.

In this context, *network password* is sometimes used as a synonym for *Secure RPC password*

Normally, this is the only time the principal is asked to provide a password. However, if the principal's private key in the cred table was encrypted with a password that was *different* from the user's login password, `login` cannot decrypt it using the login password at login time, and thus cannot provide a decrypted private key to the keyserver. (This most often occurs when a user's private key in the cred table was encrypted with a Secure RPC password different from the user's login password.)

To temporarily remedy this problem, the principal must perform a `keylogin`, using the `keylogin` command, after every login. (The `-r` flag is used to `keylogin` the superuser principal and to store the superuser's key in `/etc/.rootkey` on a host.)

For a principal user

```
keylogin
```

For a principal machine (only once)

```
keylogin -r
```

Note, however, that performing an explicit `keylogin` with the original password provides only a temporary solution good for the current login session only. The private key in the cred table is still encrypted with a password different than the user's login password so the *next* time the user logs in the problem will reoccur. To permanently solve this problem, the user must run `chkey` to change the password used to encrypt the private key to the user's login password (see "Changing Keys for a NIS+ Principal" on page 100).

Changing Keys for a NIS+ Principal

The `chkey` command changes an NIS+ principal's public and private keys that are stored in the cred table. It does not affect the principal's entry either in the `passwd` table or in the `/etc/passwd` file.

The `chkey` command:

- Generates new keys and encrypts the private key with the password. If run with the `-p` option, `chkey` re-encrypts the existing private key with a new password.

- Generates a new Diffie-Hellman key pair and encrypts the private key with the password you provide. However, in most cases you do not want a new keypair, you want to re-encrypt your *current* existing private key with the new password. To do this, you must use the `-p` flag: `chkey -p`.

See the man pages for more information on these subjects.

Note – In a NIS+ environment, when you change your login password with any of the current administration tools or the `passwd` (or `nisp passwd`) commands, your private key in the cred table is automatically re-encrypted with the new password for you. Thus, you do not need to explicitly run `chkey` after a change of login password.

The `chkey` command interacts with the keyserver, the cred table, and the `passwd` table. In order to run `chkey`, you:

- Must have an entry in the `passwd` table of your home domain. Failure to meet this requirement will result in an error message.
- Must run `keylogin` to make sure that the keyserver has a decrypted private key for you.
- Must have modify rights to the cred table. If you do not have modify rights you will get a “permission denied” type of error message.
- Must know the original password with which the private key in the cred table was encrypted. (In most cases, this your Secure RPC password.)

To use the `chkey` command to re-encrypt your private key with your login password, you first run `keylogin` using the original password, and then use `chkey -p` as shown in Table 6-1 on page 102 which illustrates how to perform a `keylogin` and `chkey` for a principal user:

Table 6-1 Re-encrypting Your Private Key : Command Summary

Tasks	Commands
Log in. Provide login password.	Sirius% login <i>Login-name</i> Password:
If login password and Secure RPC password are different, perform a keylogin.	Sirius% keylogin
Provide the original password that was used to encrypt the private key.	Password: <i>Secure RPC password</i>
Run chkey.	Sirius% chkey -p Updating nisplus publickey database Updating new key for 'unix.1199@Wiz.Com'.
Enter login password. Re-enter login. password	Enter login password: <i>login-password</i> Retype password: Done

Changing the Keys

The following sections describe how to change the keys of an NIS+ principal.

Changing Root Keys From Root

Table 6-2 on page 103 shows how to change the keys for the root master server from the root master (as root):

Table 6-2 Changing a Root Master's Keys: Command Summary

Tasks	Commands
Create new DES credentials	rootmaster# nisaddcred des
Find the Process ID of <code>rpc.nisd</code> Kill the NIS+ daemon	rootmaster# ps -e grep rpc.nisd rootmaster# kill pid
Restart NIS+ daemon with no security	rootmaster# rpc.nisd -s0
Perform a keylogout (previous keylogin is no out of date).	rootmaster# keylogout -f
Update the keys in the directories served by the master	rootmaster# nisupdkeys dirs
Find the Process ID of <code>rpc.nisd</code> Kill the NIS+ daemon	rootmaster# ps -e grep rpc.nisd rootmaster# kill pid
Restart NIS+ daemon with default security	rootmaster# rpc.nisd
Perform a keylogin	rootmaster# keylogin

Where:

- *pid* is the process ID number reported by the `ps -e | grep rpc.nisd` command.
- *dirs* are the directory objects you wish to update. (That is, the directory objects that are served by rootmaster.)

In the first step of the process outlined in Table 6-2, `nisaddcred` updates the cred table for the root master, updates `/etc.rootkey` and performs a `keylogin` for the root master. At this point the directory objects served by the master have not been updated and their credential information is now out of synch with the root master. The subsequent steps described in Table 6-2 are necessary to successfully update all the objects.

Changing Root Keys From Another Machine

To change the keys for the root master server from some other machine you must have the required NIS+ credentials and authorization to do so.

Table 6-3 Remotely Changing Root Master Keys: Command Summary

Tasks	Commands
Create the new DES credentials.	othermachine% nisaddcred -p principal -P nisprincipal des
Update the directory objects.	othermachine% nisupdkeys dirs
Update /etc.rootkey.	othermachine% keylogin -r
Reinitialize othermachine as client	othermachine% nisinit -cH

Where:

- *principal* is the root machine's Secure RPC netname. For example: `unix.rootmaster@wiz.com` (no dot at the end).
- *nis_principal* is the root machine's NIS+ principal name. For example, `rootmaster.wiz.com`. (a dot at the end).
- *dirs* are the directory objects you wish to update (that is, the directory objects that are served by `rootmaster`).

When running `nisupdkeys` be sure to update all relevant directory objects at the same time. In other words, do them all with one command. Separate updates may result in an authentication error.

Changing the Keys of a Root Replica from the Replica

To change the keys of a root replica from the replica, use these commands:

Table 6-4 Changing Keys of a Root Replica

```

replica# nisaddcred des
replica# nisupdkeys dirs

```

Where:

- *dirs* are the directory objects you wish to update, (that is, the directory objects that are served by `replica`).

When running `nisupdkeys` be sure to update all relevant directory objects at the same time. In other words, do them all with one command. Separate updates may result in an authentication error.

Changing the Keys of a Nonroot Server

To change the keys of a nonroot server (master or replica) from the server, use these commands:

Table 6-5 Changing Keys of a Root Replica

```
subreplica# nisaddcred des
subreplica# nisupdkeys parentdir dirs
```

Where:

- *parentdir* is the non-root server's parent directory (that is, the directory containing subreplica's NIS+ server).
- *dirs* are the directory objects you wish to update (that is, the directory objects that are served by subreplica).

When running `nisupdkeys` be sure to update all relevant directory objects at the same time. In other words, do them all with one command. Separate updates may result in an authentication error.

Updating Public Keys

The public keys of NIS+ servers are stored in several locations throughout the namespace. When new credential information is created for the server, a new key pair is generated and stored in the cred table. However, namespace directory objects still have copies of the server's *old* public key. The `nisupdkeys` command is used to update those directory object copies.

The `nisupdkeys` Command

If a new keypair is generated because the old key pair has been compromised or the password used to encrypt the private key is forgotten, the `nisupdkeys` can be used to update the old public key in the directory objects.

The `nisupdkeys` command can:

- Update the key of one particular server
- Update the keys of all the servers that support a NIS+ directory object
- Remove a server's public key from the directory object
- Update a server's IP address, if that has changed

However, `nisupdkeys` cannot update the `NIS_COLD_START` files on the principal workstations. To update their copies of a server's keys, NIS+ clients should run the `nisclient` command. Or, if the NIS+ cache manager is running and more than one server is available in the coldstart file, the principals can wait until the time-to-live expires on the directory object. When that happens, the cache manager automatically updates the cold-start file. The default time-to-live is 12 hours.

To use the `nisupdkeys` command, you must have modify rights to the NIS+ directory object.

Updating Public Keys Arguments and Examples

The `nisupdkeys` command is located in `/usr/lib/nis`. The `nisupdkeys` command uses the following arguments (for a complete description of the `nisupdkeys` command and a full list of all its arguments, see the `nisupdkeys` man page):

Table 6-6 `nisupdkeys` Arguments

Argument	Effect
(no argument)	Updates all keys of servers for current domain
<i>directoryname</i>	Updates the keys of the directory object for the named directory.
<code>-H servername</code>	Updates the keys of the named server for the current domain directory object. A fully qualified host name can be used to update the keys of servers in other domains.
<code>-s -H servername</code>	Updates the keys of all the directory objects served by the named server.
<code>-C</code>	Clears the keys.

Table 6-7 on page 107 gives an example of updating a public key:

Table 6-7 Updating a Public Key: Command Summary

Tasks	Commands
Update all keys of all servers of the current domain (Wiz.Com).	<pre>rootmaster# /usr/lib/nis/nisupdkeys Fetch Public key for server rootmaster.Wiz.Com. netname='unix.rootmaster@Wiz.Com' Updating rootmaster.Wiz.Com.'s public key. Public key: <i>public-key</i></pre>
Update keys of all servers supporting the Sales.Wiz.Com domain directory object.	<pre>salesmaster# nisupdkeys Sales.Wiz.Com (Screen notices not shown)</pre>
Update keys for a server named <code>server7</code> in all the directories that store them.	<pre>rootmaster# nisupdkeys -H server7</pre>
Clear the keys stored by the Sales.Wiz.Com directory object.	<pre>rootmaster# nisupdkeys -C Sales.Wiz.Com</pre>
Clear the keys for the current domain directory object for the server named <code>server7</code> .	<pre>rootmaster# nisupdkeys -C -H server7</pre>

Updating IP Addresses

If you change a server's IP address, or add additional addresses (multihome), you need to run `nisupdkeys` to update NIS+ address information.

To update the IP addresses of one or more servers, use the `nisupdkeys` command `-a` option.

To update the IP addresses of servers of a given domain

```
rootmaster# nisupdkeys -a domain
```

To update the IP address of a particular server

```
rootmaster# nisupdkeys -a -H server
```


Administering NIS+ Access Rights



This chapter assumes that you have an adequate understanding of the NIS+ security system in general, and in particular of the role that access rights play in that system (see Chapter 4, “Security Overview,” for this information).

Some NIS+ security tasks can be performed more easily with Solstice AdminSuite tools if you have them available.

This chapter provides the following general information about access rights:

<i>Introduction to Authorization and Access Rights</i>	<i>page 110</i>
<i>Concatenation of Access Rights</i>	<i>page 111</i>
<i>How Access Rights Are Assigned and Changed</i>	<i>page 112</i>
<i>Table, Column, and Entry Security</i>	<i>page 113</i>
<i>Where Access Rights Are Stored</i>	<i>page 117</i>
<i>Viewing an NIS+ Object’s Access Rights</i>	<i>page 118</i>
<i>Default Access Rights</i>	<i>page 119</i>
<i>How a Server Grants Access Rights to Tables</i>	<i>page 119</i>
<i>Specifying Access Rights in Commands</i>	<i>page 120</i>
<i>Displaying NIS+ Defaults—The nisdefaults Command</i>	<i>page 124</i>

This chapter then describes how to use the NIS+ access rights administration commands to perform the following tasks:

<i>Setting Default Security Values</i>	<i>page 125</i>
<i>Specifying Nondefault Security Values at Creation Time</i>	<i>page 127</i>
<i>Changing Object and Entry Access Rights</i>	<i>page 128</i>

<i>Specifying Column Access Rights</i>	<i>page 130</i>
<i>Changing Ownership of Objects and Entries</i>	<i>page 132</i>
<i>Changing an Object or Entry's Group</i>	<i>page 134</i>

Introduction to Authorization and Access Rights

See “NIS+ Authorization and Access—Introduction” on page 63 and Chapter 4, “Security Overview” for a description of how authorization and access rights work with NIS+ credentials and authentication to provide security for the NIS+ namespace.

Authorization Classes—Review

As described more fully in “Authorization Classes” on page 63, NIS+ access rights are assigned on a class basis. There are four different NIS+ classes:

- *Owner*. The owner class is a *single* NIS+ principal. By default, an object's owner is the principal that created the object. However, an object's owner can transfer ownership to another principal who then becomes the new owner.
- *Group*. The group class is a *collection* of one or more NIS+ principals. An NIS+ object can have only one NIS+ group.
- *World*. The world class contains all NIS+ principals that are authenticated by NIS+ (in other words, everyone in the owner and group class, plus everyone else who presents a valid DES credential).
- *Nobody*. The nobody class is composed of anyone who is not properly authenticated (in other words, anyone who does not present a valid DES credential).

Access Rights—Review

As described more fully in “NIS+ Access Rights” on page 67, there are four types of NIS+ access rights:

- *Read*. A principal with read rights to an object can view the contents of that object.
- *Modify*. A principal with modify rights to an object can change the contents of that object.

- *Destroy.* A principal with Destroy rights to an object can delete the object.
- *Create.* A principal with create rights to a higher level object can create new objects within that level. In other words, if you have create rights to a NIS+ directory object, you can create new tables within that directory. If you have create rights to a NIS+ table, you can create new columns and entries within that table.

Keep in mind that these rights logically evolve down from directory to table to table column and entry levels. For example, to create a new table, you must have create rights for the NIS+ directory object where the table will be stored. When you create that table, you become its default owner. As owner, you can assign yourself create rights to the table which allows you to create new entries in the table. If you create new entries in a table, you become the default owner of those entries. As table owner, you can also grant table level create rights to others. For example, you can give your table's group class table level create rights. In that case, any member of the table's group can create new entries in the table. The individual member of the group who creates a new table entry becomes the default owner of that entry.

Concatenation of Access Rights

Authorization classes are concatenated. In other words, the higher class usually belongs to the lower class and automatically gets the rights assigned to the lower class. It works like this:

- *Owner class.* An object's owner may, or may not, belong to the object's group. If the owner does belong to the group, then the owner gets whatever rights are assigned to the group. The object's owner automatically belongs to the world and nobody classes, so the owner automatically gets whatever rights that object assigns to those two classes.
- *Group class.* Members of the object's group automatically belong to the world and nobody classes, so the group members automatically get whatever rights that object assigns to world and nobody.
- *World class.* The world class automatically gets the same rights to an object that are given to the nobody class.
- *Nobody class.* The nobody class only gets those rights an object specifically assigns to the nobody class.

The basic principle that governs this is that access rights override the absence of access rights. In other words, a higher class can have *more* rights than a lower class, but not *fewer* rights. (The one exception to this rule is that if the owner is not a member of the group, it is possible to give rights to the group class that the owner does not have.)

How Access Rights Are Assigned and Changed

When you create an NIS+ object, NIS+ assigns that object a default set of access rights for the owner and group classes. By default, the owner is the NIS+ principal who creates the object. The default group is the group named in the `NIS_GROUP` environment variable. (See “Default Access Rights” on page 119 for details.)

Specifying Different Default Rights

NIS+ provides two different ways to change the default rights that are automatically assigned to an NIS+ object when it is created.

- The `NIS_DEFAULTS` environment variable. `NIS_DEFAULTS` stores a set of security-related default values, one of which is access rights. These default access rights are the ones automatically assigned to an object when it is created. (See “Displaying NIS+ Defaults—The `nisdefaults` Command” on page 124 for details.)

If the value of the `NIS_DEFAULTS` environment variable is changed, objects created after the change are assigned the new values. However, previously created objects are not affected.

- The `-D` option, which is available with several NIS+ commands. When you use the `-D` option as part of the command to create an NIS+ object, it overrides the default rights specified by the `NIS_DEFAULTS` environment variable and allows you to explicitly specify an initial set of rights for that object. (See “Specifying Nondefault Security Values at Creation Time” on page 127 for details.)

Changing Access Rights to an Existing Object

When an NIS+ object is created, it comes into existence with a default set of access rights (from either the `NIS_DEFAULTS` environment variable or as specified with the `-D` option). These default rights can be changed with the

- `nischmod` command
- `nistbladm` command for table columns

Table, Column, and Entry Security

NIS+ tables allow you to specify access rights on the table three ways: you can specify access rights to the table as a whole, to each table column individually, and to each entry (row) by itself. (A field is the intersection between a column and an entry (row) as shown by darker shading in the table below. All data values are entered in fields.)

Table 7-1 Table Entries and Columns

		Column	
Entry		FIELD	

These column- and entry level access rights allow you to specify *additional* access to individual rows and columns that override table level restrictions, but column and entry level rights cannot be *more* restrictive than the table as a whole:

Remember that authorization classes concatenate. Higher class gets the rights assigned to the lower class. (“Concatenation of Access Rights” on page 111 for details.)

- **Table.** The table level is the base level. Access rights assigned at the table level apply to every piece of data in the table unless specifically modified by a column or entry exception. Thus, the table level rights should be the *most* restrictive.
- **Column.** Column-level rights allow you to grant additional access rights on a column-by-column basis. For example, suppose the table level granted no access rights whatsoever to the world and nobody classes. In such a case, no one in those two classes could read, modify, create, or destroy any data in the table. You could use column-level rights to override that table level restriction and permit members of the world class the right to view data in a particular column.

On the other hand, if the table level grants table-wide read rights to the owner and group classes, you cannot use column-level rights to prevent the group class from having read rights to that column.

Keep in mind that a column's group does not have to be the same as the table's group or an entry's group. They can all have different groups.

- *Entry (row)*. entry level rights allow you to grant additional access rights on a row-by-row basis. For example, this allows you to permit individual users to change entries that apply to them, but not entries that apply to anyone else.

Keep in mind that an entry's group does not have to be the same as the table's group or a column's group. They can all have different groups. This means that you can permit members of a particular group to work with one set of entries while preventing them from affecting entries belonging to other groups.

Table, Column, Entry Example

Column- or entry level access rights can provide additional access in two ways: by extending the rights to additional principals or by providing additional rights to the same principals. Of course, both ways can be combined. Following are some examples.

Assume a table object granted read rights to the table's owner:

Table 7-2 Table, Column, Entry Example 1

	Nobody	Owner	Group	World
Table Access Rights:	----	r--	----	----

This means that the table's owner could read the contents of the entire table but no one else could read anything. You could then specify that Entry-2 of the table grant read rights to the group class:

Table 7-3 Table, Column, Entry Example 2

	Nobody	Owner	Group	World
Table Access Rights:	----	r--	----	----
Entry-2 Access Rights:	----	----	r--	----

Although only the owner could read all the contents of the table, any member of the table's group could read the contents of that particular entry. Now, assume that a particular column granted read rights to the world class:

Table 7-4 Table, Column, Entry Example 3

	Nobody	Owner	Group	World
Table Access Rights:	----	r---	----	----
Entry-2 Access Rights:	----	----	r---	----
Column-1 Access Rights:	----	----	----	r---

Members of the world class could now read that column *for all entries* in the table (light shading in Table 7-5). Members of the group class could read everything in Column-1 (because members of the group class are also members of the world class) and also all columns of Entry-2 (dark shading in Table 7-5). Neither the world nor the group classes could read any cells marked *NP* (for Nor Permitted).

Table 7-5 Table, Column, Entry Example 4

	Col 1	Col 2	Col 2
Entry-1	contents	*NP*	*NP*
Entry-2	contents	contents	contents
Entry-3	contents	*NP*	*NP*
Entry-4	contents	*NP*	*NP*
Entry-5	contents	*NP*	*NP*

Rights at Different Levels

This section describes how the four different access rights (read, create, modify, and destroy) work at the four different access levels (directory, table, column, and entry).

The objects that these various rights and levels act on are summarized in the table Table 7-6:

Table 7-6 Access Rights and Levels and the Objects They Act Upon

	Directory	Table	Column	Entry
Read	List directory contents	View table contents	View column contents	View entry (row) contents
Create	Create new directory or table objects	Add new entries (rows)	Enter new data values in a column	Enter new data values in an entry (row)
Modify	Move objects and change object names	Change data values anywhere in table	Change data values in a column	Change data values in an entry (row)
Destroy	Delete directory objects such as tables	Delete entries (rows)	Delete data values in a column	Delete data values in an entry (row)

Read Rights

- *Directory.* If you have read rights to a directory, you can list the contents of the directory.
- *Table.* If you have read rights to a table, you can view all the data in that table.
- *Column.* If you have read rights to a column, you can view all the data in that column.
- *Entry.* If you have read rights to an entry, you can view all the data in that entry.

Create Rights

- *Directory.* If you have create rights at the directory level, you can create new objects in the directory such as new tables.
- *Table.* If you have create rights at the table level, you can create new entries. (You cannot add new columns to an existing table regardless of what rights you have.)

- *Column.* If you have create rights to a column, you can enter new data values in the fields of that column. You cannot create new columns.
- *Entry.* If you have create rights to an entry, you can enter new data values in the fields of that row. (Entry level create rights do not permit you to create new rows.)

Modify Rights

- *Directory.* If you have modify rights at the directory level, you can move or rename directory objects.
- *Table.* If you have modify rights at the table level, you can change any data values in the table. You can create (add) new rows, but you cannot create new columns. If an existing field is blank, you can enter new data in it.
- *Column.* If you have modify rights to a column, you can change the data values in the fields of that column.
- *Entry.* If you have modify rights to an entry, you can change the data values in the fields of that row.

Destroy Rights

- *Directory.* If you have destroy rights at the directory level, you can destroy existing objects in the directory such as tables.
- *Table.* If you have destroy rights at the table level, you can destroy existing entries (rows) in the table but not columns. You cannot destroy existing columns in a table: you can only destroy entries.
- *Column.* If you have destroy rights to a column, you can destroy existing data values in the fields of that column.
- *Entry.* If you have destroy rights to an entry, you can destroy existing data values in the fields of that row.

Where Access Rights Are Stored

An object's access rights are specified and stored as part of the object's definition. This information is not stored in an NIS+ table.

Viewing an NIS+ Object's Access Rights

The access rights can be viewed by using the `niscat` command:

```
niscat -o objectname
```

Where *objectname* is the name of the object whose access rights you want to view.

This command returns the following information about an NIS+ object:

- *Owner*. The single NIS+ principal who has ownership rights. This is usually the person who created the object, but it could be someone to whom the original owner transferred ownership rights.
- *Group*. The object's NIS+ group.
- *Nobody class access rights*. The access rights granted to everyone, whether they are authenticated (have a valid DES credential) or not.
- *Owner class access rights*. The access rights granted to the object's owner.
- *Group class access rights*. The access rights granted to the principals in the object's group.
- *World class access rights*. The access rights granted to all authenticated NIS+ principals.

Access rights for the four authorization classes are displayed as a list of 16 characters, like this:

```
r---rmdr---r---
```

Each character represents a type of access right:

- `r` represents read rights.
- `m` represents modify rights.
- `d` represents destroy rights.
- `c` represents create rights.
- `-` represents no access rights.

The first four characters represent the access rights granted to nobody, the next four to the owner, the next four to the group, and the last four to the world:

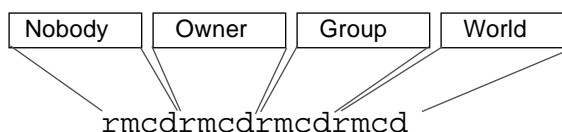


Figure 7-1 Access Rights Display

Note – Unlike UNIX file systems, the first set of rights is for nobody, not for the owner.

Default Access Rights

When you create an object, NIS+ assigns the object a default owner and group, and a default set of access rights for all four classes. The default owner is the NIS+ principal who creates the object. The default group is the group named in the `NIS_GROUP` environment variable. Initially, the default access rights are:

Table 7-7 Default Access Rights

Nobody	Owner	Group	World
-	read	read	read
-	modify	-	-
-	create	-	-
-	destroy	-	-

If you have the `NIS_DEFAULTS` environment variable set, the values specified in `NIS_DEFAULTS` will determine the defaults that are applied to new objects. When you create an object from the command line, you can use the `-D` flag to specify values other than the default values.

How a Server Grants Access Rights to Tables

This section discusses how a server grants access to tables objects, entries, and columns during each type of operation: read, modify, destroy, and create.

Note – At security level 0, a server enforces no NIS+ access rights and all clients are granted full access rights to the table object. Security level 0 is only for administrator setup and testing purposes. Do not use level 0 in any environment where ordinary users are performing their normal work.

- The four factors that a server must consider when deciding whether to grant access are:
- The type of operation requested by the principal
- The table, entry, or column the principal is trying to access
- The authorization class the principal belongs to for that particular object
- The access rights that the table, entry, or column has assigned to the principal's authorization class

After authenticating the principal making the request by making sure the principal has a valid DES credential, an NIS+ server determines the type of operation and the object of the request.

- *Directory.* If the object is a directory or group, the server examines the object's definition to see what rights are granted to the four authorization classes, determines which class the principal belongs to, and then grants or denies the request based on the principal's class and the rights assigned to that class.
- *Table.* If the object is a table, the server examines the table's definition to see what table level rights are granted to the four authorization classes, and determines which class the principal belongs to. If the class to which the principal belongs does not have table level rights to perform the requested operation, the server then determines which row or column the operation concerns and determines if there are corresponding row- or column-level access rights permitting the principal to perform the requested operation.

Specifying Access Rights in Commands

This section assumes an NIS+ environment running at security level 2 (the default).

This section describes how to specify access rights, as well as owner, group owner, and object, when using any of the commands described in this chapter.

Syntax for Access Rights

This subsection describes the access rights syntax used with the various NIS+ commands that deal with authorization and access rights.

Class, Operator, and Rights Syntax

Access rights, whether specified in an environment variable or a command, are identified with three types of arguments: *class*, *operator*, and *right*.

- *Class*. Class refers to the type of NIS+ principal (authorization class) to which the *rights* will apply.

Table 7-8 Access Rights Syntax—Class

Class	Description
n	Nobody: all unauthenticated requests
o	The owner of the object or table entry
g	The group owner of the object or table entry
w	World: all authenticated principals
a	All: shorthand for owner, group, and world (this is the default)

- *Operator*. The operator indicates the kind of operation that will be performed with the *rights*.

Table 7-9 Access Rights Syntax—Operator

Operator	Description
+	Adds the access rights specified by <i>right</i>
-	Revokes the access rights specified by <i>right</i>
=	Explicitly changes the access rights specified by <i>right</i> ; in other words, revokes all existing rights and replaces them with the new access rights.

- *Rights.* The rights are the access rights themselves. The accepted values for each are listed below.

Table 7-10 Access Rights Syntax—Rights

Right	Description
r	Reads the object definition or table entry
m	Modifies the object definition or table entry
c	Creates a table entry or column
d	Destroys a table entry or column

You can combine operations on a single command line by separating each operation from the next with a comma (,).

Table 7-11 Class, Operator, and Rights Syntax—Examples

Operations	Syntax
Add read access rights to the <i>owner</i> class	o+r
Change owner, group, and world classes' access rights to modify only from whatever they were before	a=m
Add read and modify rights to the world and nobody classes	wn+m
Remove all four rights from the group, world, and nobody classes	gwn-rmcd
Add create and destroy rights to the owner class and add read and modify rights to the world and nobody classes	o+cd,wn+rm

Syntax for Owner and Group

- *Owner.* To specify an owner, use an NIS+ principal name.
- *Group.* To specify an NIS+ group, use an NIS+ group name with the domain name appended.

For owner

principalname

Remember that by definition, principal names are fully qualified (*principalname.domainname*)

For group

```
groupname.domainname
```

Syntax for Objects and Table Entries

Objects and table entries use different syntaxes.

- Objects use simple object names.
- Table entries use indexed names.

For objects

```
objectname
```

For table entries

```
[ columnname=value ] , tablename
```

Note – In this case, the brackets are part of the syntax.

Indexed names can specify more than one column-value pair. If so, the operation applies only to the entries that match *all* the column-value pairs. The more column-value pairs you provide, the more stringent the search.

For example:

Table 7-12 Object and Table Entry—Examples

Type	Example
Object	hosts.org_dir.sales.wiz.com.
Table entry	`[uid=33555],passwd.org_dir.Eng.wiz.com.`
Two-value table entry	`[name=sales,gid=2],group.org_dir.wiz.com.`

Columns use a special version of indexed names. Because you can only work on columns with the `nistbladm` command, see “The `nistbladm` Command” on page 208 for more information.

Displaying NIS+ Defaults—The `nisdefaults` Command

The `nisdefaults` command displays the seven default values currently active in the namespace. These default values are either

- Preset values supplied by the NIS+ software
- The defaults specified in the `NIS_DEFAULTS` environment variable (if you have `NIS_DEFAULTS` values set)

Any object that you create on this machine will automatically acquire these default values unless you override them with the `-D` option of the command you are using to create the object.

Table 7-13 The Seven NIS+ Default Values and `nisdefaults` Options

Default	Option	From	Description
Domain	-d	<code>/etc/defaultdomain</code>	Displays the home domain of the workstation from which the command was entered.
Group	-g	<code>NIS_GROUP</code> environment variable	Displays the group that would be assigned to the next object created from this shell.
Host	-h	<code>uname -n</code>	Displays the workstation's host name.
Principal	-p	<code>gethostbyname()</code>	Displays the fully qualified user name or host name of the NIS+ principal who entered the <code>nisdefaults</code> command.
Access Rights	-r	<code>NIS_DEFAULTS</code> environment variable	Displays the access rights that will be assigned to the next object or entry created from this shell. Format: <code>----rmdr---r---</code>
Search path	-s	<code>NIS_PATH</code> environment variable	Displays the syntax of the search path, which indicate the domains that NIS+ will search through when looking for information. Displays the value of the <code>NIS_PATH</code> environment variable if it is set.
Time-to-live	-t	<code>NIS_DEFAULTS</code> environment variable	Displays the time-to-live that will be assigned to the next object created from this shell. The default is 12 hours.
All (terse)	-a		Displays all seven defaults in terse format.
Verbose	-v		Display specified values in verbose mode.

You can use these options to display all default values or any subset of them:

- To display all values in verbose format, type the `nisdefaults` command without arguments.

```
master% nisdefaults
Principal Name : topadmin.wiz.com.
Domain Name   : Wiz.com.
Host Name     : rootmaster.wiz.com.
Group Name    : salesboss
Access Rights  : ----rmcdr---r---
Time to live   : 12:00:00:00:00
Search Path   : Wiz.com.
```

- To display all values in terse format, add the `-a` option.
- To display a subset of the values, use the appropriate options. The values are displayed in terse mode. For example, to display the rights and search path defaults in terse mode, type:

```
rootmaster% nisdefaults -rs
----rmcdr---r---
Wiz.com.
```

- To display a subset of the values in verbose mode, add the `-v` flag.

Setting Default Security Values

This section describes how to perform tasks related to the `nisdefaults` command, the `NIS_DEFAULTS` environment variable, and the `-D` option. The `NIS_DEFAULTS` environment variable specifies the following default values:

- Owner
- Group
- Access rights
- Time-to-live.

The values that you set in the `NIS_DEFAULTS` environment variable are the default values applied to all NIS+ objects that you create using that shell (unless overridden by using the `-D` option with the command that creates the object).

You can specify the default values (owner, group, access rights, and time-to-live) specified with the `NIS_DEFAULTS` environment variable. Once you set the value of `NIS_DEFAULTS`, every object you create from that shell will acquire those defaults, unless you override them by using the `-D` option when you invoke a command.

Displaying the Value of `NIS_DEFAULTS`

You can check the setting of an environment variable by using the `echo` command, as shown below:

```
client% echo $NIS_DEFAULTS
owner=butler:group=gamblers:access=o+rmd
```

You can also display a general list of the NIS+ defaults active in the namespace by using the `nisdefaults` command as described in “Displaying NIS+ Defaults—The `nisdefaults` Command” on page 124.

Changing Defaults

You can change the default access rights, owner, and group, by changing the value of the `NIS_DEFAULTS` environment variable. Use the environment command that is appropriate for your shell (`setenv` for `csh` or `$NIS_DEFAULTS=`, `export` for `sh` and `ksh`) with the following arguments:

- `access=right`, where *right* are the access rights using the formats described in “Specifying Access Rights in Commands” on page 120.
- `owner=name`, where *name* is the user name of the owner.
- `group=group`, where *group* is the name of the default group

You can combine two or more arguments into one line separated by colons:

```
owner=principal-name:group=group-name
```

Table 7-14 on page 127 shows some examples:

Table 7-14 Changing Defaults—Examples

Tasks	Examples
This command grants owner read access as the default access right.	<code>client% setenv NIS_DEFAULTS access=o+r</code>
This command sets the default owner to be the user <code>abe</code> whose home domain is <code>Wiz.com</code> .	<code>client% setenv NIS_DEFAULTS owner=abe.wiz.com.</code>
This command combines the first two examples on one code line.	<code>client% setenv NIS_DEFAULTS access=o+r:owner=abe.wiz.com.</code>

All objects and entries created from the shell in which you changed the defaults will have the new values you specified. You cannot specify default settings for a table column or entry; the columns and entries simply inherit the defaults of the table.

Resetting the Value of NIS_DEFAULTS

You can reset the `NIS_DEFAULTS` variable to its original values, by typing the name of the variable without arguments, using the format appropriate to your shell:

For C shell

```
client# unsetenv NIS_DEFAULTS
```

For Bourne or Korn shell

```
client$ NIS_DEFAULTS=; export NIS_DEFAULTS
```

Specifying Nondefault Security Values at Creation Time

You can specify different (that is, nondefault) access rights, owner, and group, any time that you create an NIS+ object or table entry with any of the following NIS+ commands:

- `nismkdir`—Creates NIS+ directory objects

- `nisaddent`—Transfers entries into an NIS+ table
- `nistbladm`—Creates entries in an NIS+ table

To specify security values other than the default values, insert the `-D` option into the syntax of those commands, as described in “Specifying Access Rights in Commands” on page 120.

As when setting defaults, you can combine two or more arguments into one line. Remember that column and entry’s owner and group are always the same as the table, so you cannot override them.

For example, to use the `nismkdir` command to create a `sales.wiz.com` directory and override the default access right by granting the owner only read rights you would type:

```
client% nismkdir -D access=o+r sales.wiz.com
```

Changing Object and Entry Access Rights

The `nischmod` command operates on the access rights of an NIS+ object or table entry. It does not operate on the access rights of a table column; for columns, use the `nistbladm` command with the `-D` option. For all `nischmod` operations, you must already have modify rights to the object or entry.

Using `nischmod` to Add Rights

To add rights for an object or entry use:

For object

```
nischmod class+right object-name
```

For table entry

```
nischmod class+right [column-name=value],table-name
```

For example, to add read and modify rights to the group of the sales.wiz.com. directory object you would type:

```
client% nischmod g+rm sales.wiz.com.
```

For example to add read and modify rights to group for the name=abe entry in the hosts.org_dir.wiz.com. table you would type:

```
client% nischmod g+rm `[name=abe],hosts.org_dir.wiz.com.'
```

|
|
|
access-rights
entry
table

Figure 7-2 Adding Rights to a Table Entry, Example

Using nischmod to Remove Rights

To remove rights for an object or entry use:

For object

```
nischmod class-right object-name
```

For entry

```
nischmod class-right [column-name=value] , table-name
```

For example, to remove create and destroy rights from the group of the sales.wiz.com. directory object you would type:

```
client% nischmod g-cd sales.wiz.com.
```

For example to remove destroy rights from group for the name=abe entry in the hosts.org_dir.wiz.com. table, you would type.

```
client% nischmod g-d `[name=abe],hosts.org_dir.wiz.com.'
```

access-rights
entry
table

Figure 7-3 Removing Rights to a Table Entry, Example

Specifying Column Access Rights

The `nistbladm` command performs a variety of operations on NIS+ tables. Most of these tasks are described in “The `nistbladm` Command” on page 208. However, two of its options, `-c` and `-u`, enable you to perform some security-related tasks:

- *The `-c` option.* The `-c` option allows you to specify initial column access rights when creating a table with the `nistbladm` command.
- *The `-u` option.* The `-u` option allows you to change column access rights with the `nistbladm` command.

Setting Column Rights When Creating a Table

When a table is created, its columns are assigned the same rights as the table object. These table level, rights are derived from the `NIS_DEFAULTS` environment variable, or are specified as part of the command that creates the table. You can also use the `nistbladm -c` option to specify initial column access rights when creating a table with `nistbladm`. To use this option you must have create rights to the directory in which you will be creating the table. To set column rights when creating a table use:

```
nistbladm -c type `columnname=[flags] [,access]... tablename'
```

Where:

- *type* is a character string identifying the kind of table. A table's *type* can be anything you want it to be.
- *columnname* is the name of the column.

- *flags* is the type of column. Valid flags are: *S* for searchable, *I* for case insensitive, *C* for encrypted, *B* for binary data, and *X* for XDR encoded data.
- *access* is the access rights for this column that you specify using the syntax described in “Specifying Access Rights in Commands” on page 120.
- ... indicates that you can specify multiple columns each of the own type and with their own set of rights.
- *tablename* is the fully qualified name of the table you are creating.

To assign a column its own set of rights at table creation time, append access rights to each column’s equal sign after the column type and a comma. Separate the columns with a space:

```
column=type, rights column=type, rights column=type, rights
```

The example below creates a table named `depts` in the `Wiz.com` directory, of type `div`, with three columns (`Name`, `Site`, and `Manager`), and adds modify rights for the group to the second and third columns:

```
rootmaster% nistbladm -c div Name=S Site=S,g+m Manager=S,g+m depts.wiz.com.
```

For more information about the `nistbladm` and the `-c` option, see Chapter 11, “Administering NIS+ Tables.”

Adding Rights to an Existing Table Column

The `nistbladm -u` option allows you to add additional column access rights to an existing table column with the `nistbladm` command. To use this option you must have modify rights to the table column. To add additional column rights use:

```
nistbladm -u [column=access, ...], tablename
```

Where:

- *column* is the name of the column.
- *access* is the access rights for this column that you specify using the syntax described in “Specifying Access Rights in Commands” on page 120.
- ... indicates that you can specify rights for multiple columns.

- *tablename* is the fully qualified name of the table you are creating.

Use one *column=access* pair for each column whose rights you want to update. To update multiple columns, separate them with commas and enclose the entire set with square brackets:

```
[ column=access , column=access , column=access ]
```

The full syntax of this option is described in “The `nistbladm` Command” on page 208.

The example below adds read and modify rights to the group for the `name` and `addr` columns in the `hosts.org_dir.wiz.com` table.

```
client% nistbladm -u '[name=g+rm,addr=g+rm],hosts.org_dir.wiz.com.'
```

Removing Rights to a Table Column

To remove access rights to a column in an NIS+ table, you use the `-u` option as described above in “Adding Rights to an Existing Table Column” except that you subtract rights with a minus sign (rather than adding them with a plus sign).

The example below removes group’s read and modify rights to the `hostname` column in the `hosts.org_dir.wiz.com` table.

```
client% nistbladm -u 'name=g-rm,hosts.org_dir.wiz.com.'
```

Changing Ownership of Objects and Entries

The `nischown` command changes the owner of one or more objects or entries. To use it, you must have modify rights to the object or entry. The `nischown` command cannot change the owner of a column, since a table’s columns belong to the table’s owner. To change a column’s owner, you must change the table’s owner.

Changing Object Owner With `nischown`

To change an object's owner, use the following syntax:

```
nischown new-owner object
```

Where:

- *new-owner* is the fully qualified user ID of the object's new owner.
- *object* is the fully qualified name of the object.

Be sure to append the domain name to both the object name and new owner name.

The example below changes the owner of the hosts table in the Wiz.com. domain to the user named lincoln whose home domain is Wiz.com.:

```
client% nischown lincoln.wiz.com. hosts.org_dir.wiz.com.
```

Changing Table Entry Owner With `nischown`

The syntax for changing a table entry's owner uses an indexed entry to identify the entry, as shown below (this syntax is fully described in "Syntax for Objects and Table Entries" on page 123):

```
nischown new-owner [column=value, ...], tablename
```

Where:

- *new-owner* is the fully qualified user ID of the object's new owner.
- *column* is the name of the column whose value will identify the particular entry (row) whose owner is to be changed.
- *value* is the data value that identified the particular entry (row) whose owner is to be changed.
- ... indicates that you can specify ownership changes for multiple entries.

- *tablename* is the fully qualified name of the tables containing the entry whose owner is to be changed.

Be sure to append the domain name to both the new owner name and the table name.

The example below changes the owner of an entry in the Hosts table of the Wiz.com. domain to *takeda* whose home domain is Wiz.com. The entry is the one whose value in the name column is *virginia*.

```
client% nischown takeda.wiz.com. '[name=virginia],hosts.org_dir.wiz.com.'
```

Changing an Object or Entry's Group

The `nischgrp` command changes the group of one or more objects or table entries. To use it, you must have modify rights to the object or entry. The `nischgrp` command cannot change the group of a column, since the group assigned to a table's columns is the same as the group assigned to the table. To change a column's group owner, you must change the table's group owner.

Changing an Object's Group With `nischgrp`

To change an object's group, use the following syntax:

```
nischgrp group object
```

Where:

- *group* is the fully qualified name of the object's new group.
- *object* is the fully qualified name of the object.

Be sure to append the domain name to both the object name and new group name.

The example below changes the group of the hosts table in the Wiz.com. domain to `admins.wiz.com.`:

```
client% nischgrp admins.wiz.com. hosts.org_dir.wiz.com.
```


Changing a Table Entry's Group With `nischgrp`

The syntax for changing a table entry's group uses an indexed entry to identify the entry, as shown below (this syntax is fully described in "Syntax for Objects and Table Entries" on page 123):

```
nischgrp new-group [column=value, ...], tablename
```

Where:

- *new-group* is the fully qualified name of the object's new group.
- *column* is the name of the column whose value will identify the particular entry (row) whose group is to be changed.
- *value* is the data value that identified the particular entry (row) whose group is to be changed.
- *tablename* is the fully qualified name of the tables containing the entry whose group is to be changed.
- ... indicates that you can specify group changes for multiple entries.

Be sure to append the domain name to both the new group name and the table name.

The example below changes the group of an entry in the Hosts table of the Wiz.com. domain to sales.wiz.com. The entry is the one whose value in the host name column is `virginia`.

```
client% nischgrp sales.wiz.com. '[name=virginia],hosts.org_dir.wiz.com.'
```


Administering Passwords



This chapter is divided into two main parts:

You can also use AdminTool™ to change your password. You may find that more convenient than running the `passwd` command as described in this manual.

- “Using Passwords” begins on page 138 and describes how to use the `passwd` command from the point of view of an ordinary user (NIS+ principal). This section covers:

<i>Logging In</i>	<i>page 138</i>
<i>The Login incorrect Message</i>	<i>page 139</i>
<i>The password expired Message</i>	<i>page 139</i>
<i>The will expire Message</i>	<i>page 140</i>
<i>The Permission denied Message</i>	<i>page 140</i>
<i>Changing Your Password</i>	<i>page 140</i>
<i>Choosing a Password</i>	<i>page 142</i>

Some NIS+ password operations can be performed more easily with Solstice AdminSuite tools if you have them available.

- “Administering Passwords” begins on page 144 and describes how an NIS+ administrator manages the password system. This section assumes that you have an adequate understanding of the NIS+ security system in general, and in particular of the role that login passwords play in that system (see Chapter 4, “Security Overview,” for this information). This section covers:

<i>nsswitch.conf File Requirements</i>	<i>page 144</i>
<i>The nispasswd Command</i>	<i>page 144</i>
<i>The yppasswd Command</i>	<i>page 145</i>
<i>The passwd Command</i>	<i>page 145</i>

<i>The nistbladm Command</i>	<i>page 148</i>
<i>Related Commands</i>	<i>page 154</i>
<i>Displaying Password Information</i>	<i>page 154</i>
<i>Changing Passwords</i>	<i>page 156</i>
<i>Locking a Password</i>	<i>page 157</i>
<i>Managing Password Aging</i>	<i>page 158</i>
<i>Forcing Users to Change Passwords</i>	<i>page 159</i>
<i>Setting Minimum Password Life</i>	<i>page 161</i>
<i>Setting a Password Age Limit</i>	<i>page 160</i>
<i>Establishing a Warning Period</i>	<i>page 162</i>
<i>Turning Off Password Aging</i>	<i>page 164</i>
<i>Password Privilege Expiration</i>	<i>page 164</i>
<i>Specifying Maximum Number of Inactive Days</i>	<i>page 166</i>
<i>Setting Password Aging Criteria for Multiple Users</i>	<i>page 168</i>
<i>Specifying Password Criteria and Defaults</i>	<i>page 168</i>
<i>The /etc/defaults/passwd File</i>	<i>page 168</i>
<i>Password Failure Limits</i>	<i>page 171</i>

Using Passwords

When logging in to a machine, users must enter both a user name (also known as a *login ID*) and a password. Although login IDs are publicly known, passwords must be kept secret by their owners.

Logging In

Logging in to a system is a two-step process:

- 1. Type your login ID at the `Login:` prompt.**
- 2. Type your password at the `Password:` prompt.**
(To maintain password secrecy, your password is not displayed on your screen when you type it.)

If your login is successful you will see your system's message of the day (if any) and then your command-line prompt, windowing system, or normal application.

The Login incorrect Message

The `Login incorrect` message indicates that:

- You have entered the wrong login ID or the wrong password. This is the most common cause of the `Login incorrect` message. Check your spelling and repeat the process. Note that most systems limit to five the number of unsuccessful login tries you can make:
 - If you exceed a number of tries limit, you will get a `Too many failures - try later` message and not be allowed to try again until a designated time span has passed.
 - If you fail to successfully log in within a specified amount of time you will receive a `Too many tries; try again later` message, and not be allowed to try again until a designated time span has passed.
- Another possible cause of the `Login incorrect` message is that an administrator has locked your password and you cannot use it until it is unlocked. If you are sure that you are entering your login ID and password correctly, and you still get a `Login incorrect` message, contact your system administrator.
- Another possible cause of the `Login incorrect` message is that an administrator has expired your password privileges and you cannot use your password until your privileges are restored. If you are sure that you are entering your login ID and password correctly, and you still get a `Login incorrect` message, contact your system administrator.

The password expired Message

If you receive a `Your password has expired` message it means that your password has reached its age limit and expired. In other words, the password has been in use for too long and you must choose a new password at this time. (See “Choosing a Password” on page 142, for criteria that a new password must meet.)

In this case, choosing a new password is a three-step process:

- 1. Type your old password at the `Enter login password (or similar)` prompt.**

Your keystrokes are not shown on your screen.

2. **Type your new password at the `Enter new password` prompt.**
Your keystrokes are not shown on your screen.
3. **Type your new password again at the `Re-enter new password` prompt.**
Your keystrokes are not shown on your screen.

The will expire Message

If you receive a `Your password will expire in N days` message (where *N* is a number of days), or a `Your password will expire within 24 hours` message, it means that your password will reach its age limit and expire in that number of days (or hours).

In essence, this message is telling you to change your password now. (See “Changing Your Password” on page 140.)

The Permission denied Message

After entering your login ID and password, you may get a `Permission denied` message and be returned to the `login:` prompt. This means that your login attempt has failed because an administrator has either locked your password, or terminated your account, or your password privileges have expired. In these situations you cannot log in until an administrator unlocks your password or reactivates your account or privileges. Consult your system administrator.

Changing Your Password

You can also use AdminTool to change your password. You may find that more convenient than running the `passwd` command as described in this manual.

To maintain security, you should change your password regularly. (See “Choosing a Password” on page 142” for password requirements and criteria.)

Note – The `passwd` command now performs all functions previously performed by `nispasswd`. For operations specific to a NIS+ name space, use `passwd -r nisplus`.

Changing your password is a four-step process:

1. **Run the `passwd` command at a system prompt.**

2. Type your old password at the Enter login password (or similar) prompt.

Your keystrokes are not shown on your screen.

- If you receive a `Sorry: less than N days since the last change message`, it means that your old password has not been in use long enough and you will not be allowed to change it at this time. You are returned to your system prompt. Consult your system administrator to find out the minimum number of days a password must be in use before it can be changed.
- If you receive a `You may not change this password message`, it means that your network administrator has blocked any change.

3. Type your new password at the Enter new password prompt.

Your keystrokes are not shown on your screen.

At this point the system checks to make sure that your new password meets the requirements:

- If it does meet the requirements, you are asked to enter it again.
- If your new password does not meet the system requirements, a message is displayed informing you of the problem. You must then enter a new password that does meet the requirements.

See “Password Requirements” on page 142 for the requirements a password must meet.

4. Type your new password again at the Re-enter new password prompt.

Your keystrokes are not shown on your screen.

If your second entry of the new password is not identical to your first entry, you are prompted to repeat the process.

Note – When changing root’s password, you must always run `chkey -p` immediately after changing the password. (See “Changing Root Keys From Root” on page 102 and “Changing Root Keys From Another Machine” on page 104 for information on using `chkey -p` to change root’s keys.) Failure to run `chkey -p` after changing root’s password will result in root being unable to properly log in.

Password Change Failures

Some systems limit either the number of failed attempts you can make in changing your password or the total amount of time you can take to make a successful change. (These limits are implemented to prevent someone else from changing your password by guessing your current password.)

If you (or someone posing as you) fails to successfully log in or change your password within the specified number of tries or time limit, you will get a `Too many failures - try later` or `Too many tries: try again later` message. You will not be allowed to make any more attempts until a certain amount of time has passed. (That amount of time is set by your administrator.)

Choosing a Password

Many breaches of computer security involve guessing another user's password. While the `passwd` command enforces some criteria for making sure the password is hard to guess, a clever person can sometimes figure out a password just by knowing something about the user. Thus, a good password is one that is easy for you to remember but hard for someone else to guess. A bad password is one that is so hard for you to remember that you have to write it down (which you are not supposed to do), or that is easy for someone who knows about you to guess.

Password Requirements

A password must meet the following requirements:

- *Length.* By default, a password must have at least six characters. Only the first eight characters are significant. (In other words, you can have a password that is longer than eight characters, but the system only checks the first eight.) Because the minimum length of a password can be changed by a system administrator, it may be different on your system.
- *Characters.* A password must contain at least two letters (either upper- or lower-case) and at least one numeral or symbol such as `@`, `#`, `%`. For example, you can use `dog#food` or `dog2food` as a password, but you cannot use `dogfood`.

- *Not your login ID.* A password cannot be the same as your login ID, nor can it be a rearrangement of the letters and characters of your login ID. (For the purpose of this criteria, upper and lower case letters are considered to be the same.) For example, if your login ID is `Claire2` you cannot have `e2clair` as your password.
- *Different from old password.* Your new password must differ from your old one by at least three characters. (For the purpose of this criterion, upper- and lower-case letters are considered to be the same.) For example, if your current password is `Dog#f00D` you can change it to `dog#Meat` but you cannot change it to `daT#F00d`.

Bad Choices for Passwords

Bad choices for passwords include:

- Any password based on your name
- Names of family members or pets
- Car license numbers
- Telephone numbers
- Social Security numbers
- Employee numbers
- Names related to a hobby or interest
- Seasonal themes, such as Santa in December
- Any word that is in a standard dictionary

Good Choices for Passwords

Good choices for passwords include:

- Phrases plus numbers or symbols (`beam#meup`)
- Nonsense words made up of the first letters of every word in a phrase plus a number or symbol (`swotr7` for `SomeWhere Over The RainBow`)
- Words with numbers, or symbols substituted for letters (`sn00py` for `snoopy`)

Administering Passwords

Some NIS+ password operations can be performed more easily with Solstice AdminSuite tools if you have them available.

This section describes how to administer passwords in an NIS+ namespace.

Note – The `passwd` command now performs all functions previously performed by `nispasswd`. For operations specific to a NIS+ namespace, use `passwd -r nisplus`.

`nsswitch.conf` *File Requirements*

In order to properly implement the `passwd` command and password aging on your network, the `passwd` entry of the `nsswitch.conf` file on every machine must be correct. This entry determines where the `passwd` command will go for password information and where it will update password information.

Only five `passwd` configurations are permitted:

- `passwd: files`
- `passwd: files nis`
- `passwd: files nisplus`
- `passwd: compat`
- `passwd: compat`
`passwd_compat: nisplus`



Caution – All of the `nsswitch.conf` files on all of your network's workstations must use one of the `passwd` configurations shown above. If you configure the `passwd` entry in any other way, users may not be able to log in.

The nispasswd Command

All functions previously performed by the `nispasswd` command are now performed by the `passwd` command. When issuing commands from the command line, you should use `passwd`, not `nispasswd`.

Note that `nispasswd` is still retained with all of its functionality for the purpose of backward compatibility.

The yppasswd Command

All functions previously performed by the `yppasswd` command are now performed by the `passwd` command. When issuing commands from the command line, you should use `passwd`, not `yppasswd`.

Note that `yppasswd` is still retained with all of its functionality for the purpose of backward compatibility.

The passwd Command

The `passwd` command performs various operations regarding passwords. The `passwd` command replaces the `nispasswd` command. You should use the `passwd` command for all activities which used to be performed with the `nispasswd` command. (See the `passwd` command man page for a complete description of all `passwd` flags, options, and arguments.)

The `passwd` command allows users to perform the following operations:

- Change their passwords
- List their password information

Administrators can use the `passwd` command to perform the following operations:

- Force users to change their passwords the next time the log in
- Lock a user's password (prevent it from being used)
- Set a minimum number of days before a user can change passwords
- Specified when a user is warned to change passwords
- Set a maximum number of days a password can be used without being changed

`passwd` and the `nsswitch.conf` File

The name service switch determines where the `passwd` command (and other commands) obtains and stores password information. If the `passwd` entry of the applicable `nsswitch.conf` file points to:

- `nisplus`. Password information will be obtained, modified, and stored in the `passwd` and `cred` tables of the appropriate domain.
- `nis`. Password information will be obtained, modified, and stored in `passwd` maps.

- files. Password information will be obtained, modified, and stored in the `/etc/passwd` and `/etc/shadow` files.

The `passwd -r` Option

When you run the `passwd` command with the `-r nisplus`, `-r nis`, or `-r files` arguments, those options override the `nsswitch.conf` file setting. You will be warned that this is the case. If you continue, the `-r` option will cause the `passwd` command to ignore the `nsswitch.conf` file sequence and update the information in the password information storage location pointed to by the `-r` flag.

For example, if the `passwd` entry in the applicable `nsswitch.conf` file reads:

```
passwd: files nisplus
```

`files` is the first (primary) source, and `passwd` run without the `-r` option will get its password information from the `/etc/passwd` file. If you run the command with the `-r nisplus` option, `passwd` will get its information from the appropriate NIS+ `passwd` table and make its changes to that table, not to the `/etc/passwd` file.

The `-r` option should only be used when you cannot use the `nsswitch.conf` file because the search sequence is wrong. For example, when you need to update password information that is stored in two places, you can use the order specified in the `nsswitch.conf` file for the first one, but for the second one you have to force the use of the secondary or tertiary source.

The message:

```
Your specified repository is not defined in the nsswitch file!
```

indicates that your change will be made to the password information in the repository specified by the `-r` option, but that change will not affect anyone until the `nsswitch.conf` file is changed to point to that repository. For example, suppose the `nsswitch.conf` file reads `passwd: files nis` and you use the `-r nisplus` option to establish password-aging limits in a NIS+ `passwd` table. Those password-aging rules will sit in that table unused because the `nsswitch.conf` file is directing everyone to other places for their password information.

The `passwd` Command and “NIS+ Environment”

In this chapter, the phrase *NIS+ environment* refers to situations where the `passwd` entry of the applicable `nsswitch.conf` file is set to `nisplus`, or the `passwd` command is run with the `-r nisplus` argument.

The `passwd` Command and Credentials

When run in an NIS+ environment (see above), the `passwd` command is designed to function with or without credentials. Users without credentials are limited to changing their own password. Other password operations can only be performed by users who have credentials (are authenticated) and who have the necessary access rights (are authorized).

The `passwd` Command and Permissions

In this discussion of authorization and permissions, it is assumed that everyone referred to has the proper credentials.

By default, in a normal NIS+ environment the owner of the `passwd` table can change password information at any time and without constraints. In other words, the owner of the `passwd` table is normally granted full read, modify, create, and destroy authorization (permission) for that table. An owner can also:

- Assign table ownership to someone else with the `nischown` command.
- Grant some or all of read, modify, create, and destroy rights to the table’s group, or even to the world or nobody class. (Of course, granting such rights to world or nobody seriously weakens NIS+ security.)
- Change the permissions granted to any class with the `nisdefaults`, `nischmod`, or `nistbladm` commands.

Note – Regardless of what permissions they have, everyone in the world, and nobody classes are forced to comply with password-aging constraints. In other words, they cannot change a password for themselves or anyone else unless that password has aged past its minimum. Nor can members of the group, world, and nobody classes avoid having to change their own passwords when the age limit has been reached. However, age constraints do not apply to the owner of the `passwd` table.

To use the `passwd` command in an NIS+ environment, you must have the required authorization (access rights) for the operation:

Table 8-1 Access Rights for `passwd` Command

This Operation	Requires These Rights	To This Object
Displaying information	read	passwd table entry
Changing Information	modify	passwd table entry
Adding New Information	modify	passwd table

The `passwd` Command and Keys

If you use `passwd` in a NIS+ environment to change a principal's password, it tries to update the principal's private (secret) key in the cred table.

- If you have modify rights to the DES entry in the cred table and if the principal's login and Secure RPC passwords are the same, `passwd` will update the private key in the cred table.
- If you do not have modify rights to the DES entry in the cred table or if the principal's login and Secure RPC passwords are not the same, the `passwd` command will change the password, but not change the private key.

If you do not have modify rights to the DES entry, it means that the private key in the cred table will have been formed with a password that is now different from the one stored in the passwd table. In this case, the user will have to change keys with the `chkey` command or run `keylogin` after each login.

The `passwd` Command and Other Domains

To operate on the passwd table of another domain, use:

```
passwd [options] -D domainname
```

The `nistbladm` Command

The `nistbladm` command allows you to create, change, and display information about any NIS+ table, including the passwd table.

It is possible to use the `nistbladm` command to:

- Create new passwd table entries
- Delete an existing entry
- Change the UID and GID fields in the passwd table
- Change access rights and other security-related attributes of the passwd table
- Set expiration and inactivity periods for a user's account (see "Password Privilege Expiration" on page 164 and "Specifying Maximum Number of Inactive Days" on page 166.)
- Set password parameters for multiple users at once (see "Setting Password Aging Criteria for Multiple Users" on page 168).




Caution – To perform password operations using the `nistbladm` command you must apply `nistbladm` to the shadow column of the passwd table. Applying `nistbladm` to the shadow column is complex and tricky. Therefore, you should not use the `nistbladm` command for any operation that can more easily be performed by the `passwd` command or by using the AdminTool or Solstice AdminSuite tools. You should use the `passwd` command or Solstice AdminSuite tools to perform the following operations:

- Changing a password
- Setting the maximum period that a password can be used (password aging).
- Setting the minimum period that a password must be used.
- Setting the password warning period.
- Turning off password aging

`nistbladm` and Shadow Column Fields

You use the `nistbladm` command to set password parameters by specifying the values of the different fields in the shadow column. These fields are entered in the format:

```
nistbladm -m shadow=n1:n2:n3:n4:n5:n6:n7 [name=login],passwd.orgd_ir
```



 Lastchange Min Max Warn Inactive Expire Unused

Where:

- *N1 Lastchange*. The date of the last password change expressed as a number of days since January 1, 1970. The value in this field is automatically updated each time the user changes passwords. (See “nistbladm And the Number of Days” on page 152 for important information regarding the number of days.) If the field is blank, or contains a zero, it indicates that there has not been any change in the past.

Note that the number of days in the `lastchange` field is the base from which other fields and operations are calculated. Thus, an incorrect change in this field could have unintended consequence in regards to minimum, maximum, warning, and inactive time periods.

- *N2 Min*. The minimum number of days that must pass since the last time the password was changed before the user can change passwords again. For example, if the value in the `lastchange` field is 9201 (that is, 9201 days since 1/1/70) and the value in the `min` field is 8, the user is unable to change passwords until after day 9209. See “Setting Minimum Password Life” on page 161 for additional information on password minimums.

Where *min* is one of the following values:

- *Zero (0)*. A value of zero in this field (or a blank space) means that there is no minimum period
- *Greater than zero*. Any number greater than zero sets that number of days as the minimum password life.
- *Greater than max*. A value in this field that is greater than the value in the `max` field prevents the user from ever changing passwords. The message: `You may not change this password` is displayed when the user attempts to change passwords.
- *N3 Max*. The maximum number of days that can pass since the last time the password was changed. Once this maximum number of days is exceeded, the user is forced to choose a new password the next time the user logs in. For example, if the value in the `lastchange` field is 9201 and the value in the `max` field is 30, after day 9231 (figured $9201+30=9231$), the user is forced to choose a new password at the next login. See “Setting a Password Age Limit” on page 160 for additional information on password maximums.

Where *max* is one of the following values:

- *Zero (0)*. A value of zero (0) forces the user to change passwords the next time the user logs in, and it then turns off password aging.
- *Greater than zero*. Any number greater than zero sets that number of days before the password must be changed.

- *Minus one (-1)*. A value of minus one (-1) turns off password aging. In other words, entering `passwd -x -1 username` cancels any previous password aging applied to that user. A blank space in the field is treated as if it were a minus one.
- *N4 Warn*. The number of days before a password reaches its maximum that the user is warned to change passwords. For example, suppose the value in the `lastchange` field is 9201, the value in the `max` field is 30, and the value in the `warn` field is 5. Then after day 9226 (figured $9201+30-5=9226$) the user starts receiving “change your password” type warnings at each logging time. See “Establishing a Warning Period” on page 162 for additional information on password warning times.

Where *warn* is one of the following values:

- *Zero (0)*. No warning period.
- *Greater than zero*. A value of zero (0) sets the warning period to that number of days.
- *N5 Inactive*. The maximum number of days between logins. If this maximum is exceeded, the user is not allowed to log in. For example, if the value of this field is 6, and the user does not log in for six days, on the seventh day the user is no longer allowed to log in. See “Specifying Maximum Number of Inactive Days” on page 166 for additional information on account inactivity.

Where *inactive* is one of the following values:

- *Minus one (-1)*. A value of minus one (-1) turns off the inactivity feature. The user can be inactive for any number of days without losing login privileges. This is the default.
- *Greater than zero*. A value greater than zero sets the maximum inactive period to that number of days.
- *N6 Expire*. The date on which a password expires, expressed as a number of days since January 1, 1970. After this date, the user can no longer log in. For example, if this field is set to 9739 (September 1, 1995) on September 2, 1995 GMT, the user will not be able to login and will receive a `Login incorrect` message after each try. See “Password Privilege Expiration” on page 164 for additional information on password expiration.

Where *expire* is one of the following values:

- *Minus one (-1)*. A value of minus one (-1) turns off the expiration feature. If a user's password has already expired, changing this value to -1 restores it. If you do not want to set any expiration date, type a -1 in this field.
- *Greater than zero*. A value greater than zero sets the expiration date to that number of days since 1/1/70. If you enter today's date or earlier, you immediately deactivate the users password.
- *N7 Unused*. This field is not currently used. Values entered in this field will be ignored.
- *Login* is the user's login ID



Caution - When using `nistbladm` on the shadow column of the password table, all of the numeric fields must contain appropriate values. You cannot leave a field blank, or enter a zero, as a *no change* placeholder.

For example, to specify that the user amy last changed her password on day 9246 (MaY 1, 1995), cannot change her password until it has been in use for 7 days, must change her password after 30 days, will be warned to change her password after the 25th day, must not remain inactive more than 15 days, and has an account that will expire on day number 9255, you would type:

```
master# nistbladm -m shadow=9246:7:30:5:15:9255:0 [name=amy],passwd.org_dir
```

`nistbladm` *And the Number of Days*

Most password aging parameters are expressed in number of days. The following principles and rules apply:

- Days are counted from January 1, 1970. That is day zero. January 2, 1970, is day 1.
- NIS+ uses Greenwich mean time (GMT) in figuring and counting days. In other words, the day count changes at midnight GMT.
- When you specify a number of days, you must use a whole number. You cannot use fractions of days.

- When the number of days is used to specify some action, such as locking a password, the change takes effect on the day. For example, if you specify that a user's password privilege expires on day 9125 (January 2, 1995), that is the last day that the user can use the password. On the next day, the user can no longer use the password.

Values are entered in both the `lastchange` the `expire` fields as a number of days since January 1, 1970. For example:

Table 8-2 Number of Days Since 1/1/70

Date	Day Number
January 1, 1970	0
January 2, 1970	1
January 2, 1971	365
January 2, 1995	9125
March 1, 1995	9184
May 1, 1995	9246
July 1, 1995	9306
September 1, 1995	9369
November 1, 1995	9431
January 1, 1996	9493
March 1, 1996	9553
May 1, 1996	9615
July 1, 1996	9677
September 1, 1996	9739
November 1, 1996	9801
January 1, 1997	9863

Related Commands

You can also use Solstice AdminSuite tools to perform some password operations.

The `passwd` and `nistbladm` commands provide capabilities that are similar to those offered by other commands. Table 8-3 summarizes their differences.

Table 8-3 Related Commands

Command	Description
<code>yppasswd</code>	Is now linked to the <code>passwd</code> command. Using <code>yppasswd</code> simply invokes the <code>passwd</code> command.
<code>nispasswd</code>	Is now linked to the <code>passwd</code> command. Using <code>nispasswd</code> simply invokes the <code>passwd</code> command.
<code>niscat</code>	Can be used to display the contents of the <code>passwd</code> table.

Displaying Password Information

You can also use Solstice AdminSuite tools to display password information

You can use the `passwd` command to display password information about all users in a domain or about one particular user:

For your password information

```
passwd -s
```

For all users in current domain

```
passwd -s -a
```

For a particular user

```
passwd -s username
```

Only the entries and columns for which you have read permission will be displayed. Entries are displayed with the following format:

- *Without password aging:* `username status`

- *With password aging:* `username status mm/dd/yy min max warn where`

Table 8-4 NIS+ Password Display Format

Field	Description	For Further Information
<code>username</code>	The user's login name.	
<code>status</code>	The user's password status. PS indicates the account has a password. LK indicates the password is locked. NP indicates the account has no password.	See "Locking a Password" on page 157.
<code>mm/dd/yy</code>	The date, based on Greenwich mean time, that the user's password was last changed.	
<code>min</code>	The minimum number of days since the last change that must pass before the password can be changed again.	See "Setting Minimum Password Life" on page 161.
<code>max</code>	The maximum number of days the password can be used without having to change it.	See "Setting a Password Age Limit" on page 160.
<code>warn</code>	The number of days' notice that users are given before their passwords have to be changed.	See "Establishing a Warning Period" on page 162.
<code>expire</code>	A date on which users lose the ability to log in to their accounts.	See "Password Privilege Expiration" on page 164.
<code>inactive</code>	A limit on the number of days that an account can go without being logged in to. Once that limit is passed without a log in users can no longer access their accounts.	See "Specifying Maximum Number of Inactive Days" on page 166.

To display entries from a `passwd` table in another domain, use the `-D` option:

For all users in another domain

```
passwd -s -a -D domainname
```

For a particular user

```
passwd -s -D domainname username
```

Changing Passwords

You can also use Solstice AdminSuite tools to change your, or someone else's, password

New passwords must meet the criteria described in “Password Requirements” on page 142.

Changing Your Own Password

To change your password, type

```
station1% passwd
```

You will be prompted for your old password and then the new password and then the new password a second time to confirm it.

Changing Someone Else's Password

To change someone else's password, use:

To change another user's password in the same domain

```
passwd username
```

To change another user's password in a different domain

```
passwd -D domainname username
```

When using the `passwd` command in a NIS+ environment (see page 147) to change someone else's password you must have modify rights to that user's entry in the `passwd` table (this usually means that you are a member of the group for the `passwd` table and the group has modify rights). You do not have to enter either the user's old password or your password. You will be prompted to enter the new password twice to make sure that they match. If they do not match, you will be prompted to enter them again.

Changing Root's Password

When changing root's password, you must always run `chkey -p` immediately after changing the password with the `passwd` command. Failure to run `chkey -p` after changing root's password will result in root being unable to properly log in.

To change a root password, follow these steps:

- 1. Log in as root.**
- 2. Change root's password using `passwd`.**
Do not use `nispasswd`.
- 3. Run `chkey -p`.**
You *must* use the `-p` option.

Locking a Password

You can also use Solstice AdminSuite tools to lock a user's password.

When operating in a NIS+ environment (see page 147), an administrator (a group member) with modify rights to a user's entry in the `passwd` table can use the `passwd` command to lock a password. An account with a locked password cannot be used. When a password is locked, the user will receive a `Login incorrect` message after each login attempt.

Keep in mind that locked passwords have no effect on users who are already logged in. A locked password only prevents users from performing those operations that require giving a password such as `login`, `rlogin`, `ftp`, or `telnet`.

Note also that if a user with a locked password is already logged in, and that user uses the `passwd` command to change passwords, the lock is broken.

Security tips

You can use this feature to

- Temporarily lock a user's password while that user is on vacation or leave. This prevents anyone from logging in as the absent user.
- Immediately lock one or more user passwords in the case of suspected security problem.
- Quickly lock a fired employee out of the system. This is quicker and easier than eliminating that user's account and is an easy way of preserving any data stored in that account.

- If you have assigned passwords to UNIX processes, you can lock those passwords. This allows the process to run, but prevents anyone from logging in as those processes even if they know the process password. (In most cases, processes would not be set up as NIS+ principals, but would maintain their password information in `/etc` files. In such a case you would have to run the `passwd` command in files mode to lock `/etc` stored passwords.)

To lock a password, use:

```
passwd -l username
```

Unlocking a Password

To unlock a user's password, you simply change it. You can "change" it back to the exact same password that it was when it was locked. Or you can change it to something new.

For example, to unlock jody's password, you would type:

```
station1% passwd jody
```

Managing Password Aging

Password aging is a mechanism you can use to force users to periodically change their passwords.

Password aging allows you to:

- Force a user to choose a new password the next time the user logs in. (See "Forcing Users to Change Passwords" on page 159 for details.)
- Specify a maximum number of days that a password can be used before it has to be changed. (See "Setting a Password Age Limit" on page 160 for details.)
- Specify a minimum number of days that a password has to be in existence before it can be changed. (See "Setting Minimum Password Life" on page 161 for details.)

- Specify that a warning message be displayed whenever a user logs in a specified number of days before the user's password time limit is reached. (See "Establishing a Warning Period" on page 162 for details.)
- Specify a maximum number of days that an account can be inactive. If that number of days pass without the user logging in to the account, the user's password will be locked. (See "Specifying Maximum Number of Inactive Days" on page 166 for details.)
- Specify an absolute date after which a user's password cannot be used, thus denying the user the ability to log on to the system. (See "Password Privilege Expiration" on page 164 for details.)

Keep in mind that users who are already logged in when the various maximums or dates are reached are not affected by the above features. They can continue to work as normal.

Password aging limitations and activities are only activated when a user logs in or performs one of the following operations:

- login
- rlogin
- telnet
- ftp

These password aging parameters are applied on user-by-user basis. You can have different password aging requirements for different users. (You can also set general default password aging parameters as described in "The /etc/defaults/passwd File" on page 168.)

Forcing Users to Change Passwords

There are two ways to force a user to change passwords the next time the user logs in:

Force change keeping password aging rules in effect

```
passwd -f username
```

Force change and turn off password aging rules

```
passwd -x 0 username
```

Setting a Password Age Limit

You can also use Solstice AdminSuite tools to set this parameter for a user's password.

The `max` argument to the `passwd` command sets an age limit for the current password. In other words, it specifies the number of days that a password remains valid. After that number of days, a new password must be chosen by the user. Once the maximum number of days have passed, the next time the user tries to login with the old password a `Your password has been expired for too long` message is displayed and the user is forced to choose a new password in order to finish logging in to the system.

The `max` argument uses the following format:

```
passwd -x max username
```

Where:

- *username* is the login ID of the user
- *max* is one of the following values:
 - *Greater than zero.* Any number greater than zero sets that number of days before the password must be changed.
 - *Zero (0).* A value of zero (0) forces the user to change passwords the next time the user logs in, and it then turns off password aging.
 - *Minus one (-1).* A value of minus one (-1) turns off password aging. In other words, entering `passwd -x -1 username` cancels any previous password aging applied to that user.

For example, to force the user `schweik` to change passwords every 45 days, you would type the command:

```
station1% passwd -x 45 schweik
```

Setting Minimum Password Life

The *min* argument to the `passwd` command specifies the number of days that must pass before a user can change passwords. If a user tries to change passwords before the minimum number of days has passed, a `Sorry less than N days since the last change` message is displayed.

The *min* argument uses the following format:

```
passwd -x max -n min username
```

Where:

- *username* is the login ID of the user
- *max* is the maximum number of days a password is valid as described in the section above
- *min* is the minimum number of days that must pass before the password can be changed.

For example, to force the user `eponine` to change passwords every 45 days, and prevent him from changing it for the first 7 days you would type the command:

```
station1% passwd -x 45 -n 7 eponine
```

The following rules apply to the *min* argument:

- You do not have to use a *min* argument or specify a minimum number of days before a password can be changed.
- If you do use the *min* argument, it must always be used in conjunction with the *max* argument. In other words, in order to set a minimum value you must also set a maximum value.
- If you set *min* to be greater than *max*, the user is unable to change passwords at all. For example, the command `passwd -x 7 -n 8` prevents the user from changing passwords. If the user tries to change passwords, the `You may not change this password` message is displayed.

You can also use Solstice AdminSuite tools to set this parameter for a user's password.

Establishing a Warning Period

The *warn* argument to the `passwd` command specifies the number of days before a password reaches its age limit that users will start to seeing a `Your password will expire in N days` message (where *N* is the number of days) when they log in.

For example, if a user's password has a maximum life of 30 days (set with the *max* argument) and the *warn* value is set to 7 days, when the user logs in on the 24th day (one day past the *warn* value) the warning message `Your password will expire in 7 days` is displayed. When the user logs in on the 25th day the warning message `Your password will expire in 6 days` is displayed.

Keep in mind that the warning message is not sent by Email or displayed in a user's console window. It is displayed only when the user logs in. If the user does not log in during this period, no warning message is given.

Keep in mind that the *warn* value is *relative* to the *max* value. In other words, it is figured backwards from the deadline set by the *max* value. Thus, if the *warn* value is set to 14 days, the `Your password will expire in N days` message will begin to be displayed two weeks before the password reaches its age limit and must be changed.

Because the *warn* value is figured relative to the *max* value, it only works if a *max* value is in place. If there is no *max* value, *warn* values are meaningless and are ignored by the system.

The *warn* argument uses the following format:

```
passwd -x max -w warn username
```

Where:

- *username* is the login ID of the user.
- *max* is the maximum number of days a password is valid as described on page 160.
- *warn* is the number of days before the password reaches its age limit that the warning message will begin to be displayed.

For example, to force the user nilovna to change passwords every 45 days, and display a warning message 5 days before the password reaches its age limit you would type the command:

```
station1% passwd -x 45 -w 5 nilovna
```

The following rules apply to the *warn* argument:

- You do not have to use the *warn* argument or specify a warning message. If no *warn* value is set, no warning message is displayed prior to a password reaching its age limit.
- If you do use the *warn* argument, it must always be used in conjunction with the *max* argument. In other words, in order to set a warning value you must also set a maximum value.

Turning Off Password Aging

You can also use Solstice AdminSuite tools to set this parameter for a user's password.

There are two ways to turn off password aging for a given user:

Turn off aging while allowing user to retain current password

```
passwd -x -1 username
```

Force user to change password at next login, and then turn off aging

```
passwd -x 0 username
```

This sets the *max* value to either zero or -1 (see “Setting a Password Age Limit” on page 160 for more information on this value).

For example, to force the user mendez to change passwords the next time he logs in and then turn off password aging you would type the command:

```
station% passwd -x 0 mendez
```

You can also use the `nistbladm` command to set this value. For example, to turn off password aging for the user `otsu` and allow her to continue using her current password, you would type:

```
station1% nistbladm -m 'shadow=0:0:-1:0:0:0:0' [name=otsu],passwd.org_dir
```

For additional information on using the `nistbladm` command, see “The `nistbladm` Command” on page 148.

Password Privilege Expiration

You can set a specific date on which a user’s password privileges expires. When a user’s password privilege expires, that user can no longer have a valid password at all. In effect, this locks the user out of the system after the given date because after that date the user can no longer log in.

For example, if you specify an expire date of December 31, 1995, for a user named `petew`, on January 1, 1996 he will not be able to log in under that user ID regardless of what password he uses. After each login attempt he will receive a `Login incorrect` message.

Password Aging versus Expiration

Expiration of a user’s password privilege is not the same as password aging.

- *Password aging.* A password that has not been changed for longer than the aging time limit is sometimes referred to as an *expired password*. But that password can still be used to log in *one* more time. As part of that last login process the user is forced to choose a new password.
- *Expiration of password privilege.* When a user’s password *privilege* expires, the user cannot log in at all with *any* password.) In other words, it is the user’s permission to log in to the network that has expired.

Setting an Expiration Date

Password privilege expiration dates only take effect when the user logs in. If a user is *already* logged in, the expiration date has no affect until the user logs out or tries to use `rlogin` or `telnet` to connect to another machine at which

Security Tip

time the user will not be able to log in again. Thus, if you are going to implement password privilege expiration dates, you should require your users to log out at the end of each day's work session.

Note – If you have Solstice AdminSuite tools available, do not use `nistbladm` to set an expiration date. Use Solstice AdminSuite tools because they are easier to use and provides less chance for error.

To set an expiration date with the `nistbladm` command:

```
nistbladm -m `shadow=n:n:n:n:n6:n` [name=login],passwd.org_dir
```

Where:

- *login* is the user's login ID
- *n* indicates the values in the other fields of the shadow column.
- *n6* is the date on which the user's password privilege expires. This date is entered as a number of days since January 1, 1970 (see Table 8-2 on page 153). *n6* can be one of the following values:
 - *Minus one (-1)*. A value of minus one (-1) turns off the expiration feature. If a user's password has already expired, changing this value to -1 restores (un-expires) it. If you do not want to set any expiration date, type -1 in this field.
 - *Greater than zero*. A value greater than zero sets the expiration date to that number of days since 1/1/70. If you enter today's date or earlier, you immediately expire the user's password.

For example, to specify an expiration date for the user `petew` of December 31, 1995 you would type:

```
station1% nistbladm -m `shadow=n:n:n:n:n9493:n` [name=petew],passwd.org_dir
```



Caution – All of the fields must be filled in with valid values.

Turning Off Password Privilege Expiration

To turn off or deactivate password privilege expiration, you must use the `nistbladm` command to place a `-1` in this field. For example, to turn off privilege expiration for the user `huck`, you would type:

```
station1% nistbladm -m 'shadow=n:n:n:n:n:-1:n' [name=huck],passwd.org_dir
```

Or you can use the `nistbladm` command reset the expiration date to some day in the future by entering a new number of days in the `n6` field.

Specifying Maximum Number of Inactive Days

You can set a maximum number of days that a user can go without logging in on a given machine. Once that number of days passes without the user logging in, that machine will no longer allow that user to log in. In this situation, the user will receive a `Login incorrect` message after each login attempt.

This feature is tracked on a machine-by-machine basis, not a network-wide basis. That is, in an NIS+ environment, you specify the number of days a user can go without logging in by placing an entry for that user in the `passwd` table of the user's home domain. That number applies for that user on all machines on the network. However, the date on which a user last logged in to a given machine is maintained on a machine-by-machine basis in the machine's `/var/adm/utmp` file.

For example, suppose you specify a maximum inactivity period of 10 days for the user `samh`. On January 1, `samh` logs in to both machine-A and machine-B, and then logs off both machines. Four days later on January 4, `samh` logs in on machine-B and then logs out. Nine days after that on January 13, `samh` can still log `-n` to machine-B because only 9 days have elapsed since the last time he logged in on that machine, but he can no longer log in to machine-A because thirteen days have passed since his last log in on that machine.

Keep in mind that an inactivity maximum cannot apply to a machine the user has never logged in to. No matter what inactivity maximum has been specified or how long it has been since the user has logged in to some other machine, the user can always log in to a machine that the user has never logged in to before.



Caution – Do not set inactivity maximums unless your users are instructed to log out at the end of each workday. The inactivity feature only relates to logins; it does not check for any other type of system use. If a user logs in and then leaves the system up and running at the end of each day, that user will soon pass the inactivity maximum because there has been no login for many days. When that user finally does reboot or log out, he or she won't be able to log in.

Note – If you have Solstice AdminSuite tools available, do not use `nistbladm` to set an inactivity maximum. Use Solstice AdminSuite tools because they are easier to use and provide less chance for error.

To set a login inactivity maximum, you must use the `nistbladm` command in the format:

```
nistbladm -m 'shadow=n:n:n:n5:n:n' [name=login],passwd.org_dir
```

Where:

- *login* is the user's login ID
- *n* indicates the values in the other fields of the shadow column.
- *n5* is the number of days the user is allowed to go between logins. *Inactive* can be one of the following values:
 - *Minus one (-1)*. A value of minus one (-1) turns off the inactivity feature. The user can be inactive for any number of days without losing login privileges. This is the default.
 - *Greater than zero*. A value greater than zero sets the maximum inactive period to that number of days.

For example, to specify that the user `samh` must log in at least once every seven days, you would type:

```
station1% nistbladm -m 'shadow=n:n:n:n:7:n:n' [name=samh],passwd.org_dir
```

To clear an inactivity maximum and allow a user who has been prevented from logging in to log in again, use `nistbladm` to set the inactivity value to -1.

Setting Password Aging Criteria for Multiple Users

You can use the `nistbladm` command globally specify password *max*, *min*, *warn*, *inactive*, and *expire*, values for all principals listed in a given `passwd` table.

To globally change password aging values for all users listed in a given password table, you use the `nistbladm` command without an indexed entry between the square brackets. For example, to globally set a minimum of 7 days, a maximum of 30 days, a warning period of 5 days, and no inactivity limit or expire date you would type:

```
station1% nistbladm -m `shadow=n:7:30:5:-1:-1:0' [],passwd.org_dir
```

You can also use the `nistbladm` command to turn off password aging for all users in a given password table by globally setting their *max* value to -1 or 0 as described in “Turning Off Password Aging” on page 163.

Note – The value you enter in the `lastchange` field (the first field) will be applied to all the users. In effect, you will be resetting everyone’s last change date to that value.

Specifying Password Criteria and Defaults

The following subsections describe various password-related defaults and general criteria that you can specify.

The /etc/defaults/passwd File

The `/etc/defaults/passwd` file is used to set four general password defaults for users whose `nsswitch.conf` file points to files. The defaults set by the `/etc/defaults/passwd` file apply only to users whose operative password information is taken from `/etc` files; they do not apply to anyone using either NIS maps or NIS+ tables. An `/etc/defaults/passwd` file on an NIS+ server only affects local users who happen to be obtaining their password information from those local files. An `/etc/defaults/passwd` file on an NIS+ server has no effect on the NIS+ environment or users whose `nsswitch.conf` file points to either `nis` or `nisplus`.

The four general password defaults governed by the `/etc/defaults/passwd` file are:

- Maximum number of weeks the password is valid
- Minimum number of weeks the password is valid
- The number of weeks before the password becomes invalid that the user is warned
- The minimum number of characters that a password must contain

The following principles apply to defaults set with an `/etc/defaults/passwd` file:

- For users who obtain password information from local `/etc` files, individual password aging maximums, minimums and warnings set by the `passwd` command or Solstice AdminSuite or AdminTool override any `/etc/defaults/passwd` defaults. In other words, defaults set in the `/etc/defaults/passwd` file are only applied to those users who do not have corresponding individual settings in their entries in their `passwd` table.
- Except for password length, all the `/etc/defaults/passwd` file defaults are expressed as a number of weeks. (Remember that *individual* password aging times are expressed as a number of days.)
- The `MAXWEEKS`, `MINWEEKS`, and `WARNWEEKS` defaults are all counted forward from the date of the user's last password change. (Remember that *individual warn* values are counted backwards from the maximum date.)

By default, `/etc/defaults/passwd` files already contain the entries:

```
MAXWEEKS=  
MINWEEKS=  
PASSLENGTH=
```

To implement an entry, simply type the appropriate number after the equal sign. Entries that do not have a number after the equal sign are inactive and have no affect on any user. Thus, to set a MAXWEEKS default of 4, you would change the `/etc/defaults/passwd` file to read:

```
MAXWEEKS=4
MINWEEKS=
PASSLENGTH=
```

Maximum Weeks

You can use the MAXWEEKS default in the `/etc/defaults/passwd` file to set the maximum number of weeks that a user's password is valid. To set a default maximum time period, type the appropriate number of weeks after the equal sign on the MAXWEEKS= line:

```
MAXWEEKS=N
```

Where *N* is a number of weeks. For example, MAXWEEKS=9.

Minimum Weeks

You can use the MINWEEKS default in the `/etc/defaults/passwd` file to set the minimum number of weeks that must pass before a user can change passwords. To set a default minimum time period, type the appropriate number of weeks after the equal sign on the MINWEEKS= line:

```
MINWEEKS=N
```

Where *N* is a number of weeks. For example, MINWEEKS=2.

Warning Weeks

You can add a WARNWEEKS= default to the `/etc/defaults/passwd` file to set the number of weeks prior to a password becoming invalid due to aging that the user is warned. For example, if you have set the MAXWEEKS default to 9, and you want users to be warned two weeks before their passwords become invalid, you would set the WARNWEEKS default to 7.

Remember that WARNWEEKS are counted forward from the date of the user's last password change, not backward from the MAXWEEKS expiration date. Thus, WARNWEEKS must always be less than MAXWEEKS and cannot be equal to or greater than MAXWEEKS.

There is no point in setting a WARNWEEKS default unless you also set a MAXWEEKS default.

To set the warning time period, type the appropriate number of weeks after the equal sign on the `WARNWEEKS=` line:

```
WARNWEEKS=N
```

Where *N* is a number of weeks. For example, `WARNWEEKS=1`.

Minimum Password Length

By default, the `passwd` command assumes a minimum length of six characters. You can use the `PASSLENGTH` default in the `/etc/defaults/passwd` file to change that by setting the minimum number of characters that a user's password must contain to some other number.

To set the minimum number of characters to something other than six, type the appropriate number of characters after the equal sign on the `PASSLENGTH=` line:

```
PASSLENGTH=N
```

Where *N* is a number of characters. For example, `PASSLENGTH=7`.

Password Failure Limits

You can specify a number-of-tries limit or an amount-of-time limit (or both) for a user's attempt to change passwords. These limits are specified by adding arguments when starting the `rpc.nispasswd` daemon.

Limiting the number of attempts or setting a time frame provides a limited (but not foolproof) defense against unauthorized persons attempting to change a valid password to one that they discover through trial and error.

Maximum Number of Tries

To set the maximum number of times a user can try to change a password without succeeding, use the `-a number` argument with `rpc.nispasswd`, where *number* is the number of allowed tries. (You must have superuser privileges on the NIS+ master server to run `rpc.nispasswd`.)

For example, to limit users to no more than four attempts (the default is 3), you would type:

```
station1# rpc.nispasswd -a 4
```

Security Tip

In this case, if a user's fourth attempt at logging in is unsuccessful, the message `Too many failures - try later` is displayed. No further attempts are permitted for that user ID until a specified period of time has passed.

Maximum Login Time Period

To set the maximum amount a time a user can take to successfully change a password, use the `-c minutes` argument with `rpc.nispasswd`, where *minutes* is the number of minutes a user has to log in. (You must have superuser privileges on the NIS+ master server to run `rpc.nispasswd`.)

For example, to specify that users must successfully log in within 2 minutes, you would type:

```
station1# rpc.nispasswd -c 2
```

In this case, if a user is unable to successfully change a password within 2 minutes, the message is displayed at the end of the two-minute period. No further attempts are permitted for that user ID until a specified period of time has passed.

Administering NIS+ Groups



A NIS+ group is a set of NIS+ principals. NIS+ groups are used to assign a set of access rights to NIS+ objects to the members of the group.

This chapter describes how to use NIS+ group administration commands to perform the following tasks:

<i>Specifying Group Members</i>	<i>page 174</i>
<i>Listing the Object Properties of a Group</i>	<i>page 176</i>
<i>Creating an NIS+ Group</i>	<i>page 178</i>
<i>Deleting an NIS+ Group</i>	<i>page 179</i>
<i>Adding Members to an NIS+ Group</i>	<i>page 179</i>
<i>Listing the Members of an NIS+ Group</i>	<i>page 180</i>
<i>Removing Members From an NIS+ Group</i>	<i>page 181</i>
<i>Testing for Membership in an NIS+ Group</i>	<i>page 182</i>

For a complete description of these commands and their syntax and options, see the NIS+ man pages.

Related Commands

If you have them available, Solstice AdminSuite tools provide easier methods of performing some group-related tasks.

The `nisgrpadm` command performs most group administration tasks, but several other commands affect groups as well:

Table 9-1 Commands That Affect Groups

Command	Description	See
<code>nissetup</code>	Creates, among other things, the directory in which a domain's groups are stored: <code>groups_dir</code> .	page 223
<code>nislsls</code>	Lists the contents of the <code>groups_dir</code> directory; in other words, all the groups in a domain.	page 184
<code>nischgrp</code>	Changes or assigns a group to any NIS+ object.	page 134
<code>nisdefault s</code>	Lists, among other things, the group that will be assigned to any new NIS+ object.	page 124

For a complete description of these commands and their syntax, and options, see the NIS+ man pages.

Specifying Group Members

NIS+ groups can have three types of members: explicit, implicit, and recursive; and three types of nonmembers, also explicit, implicit, and recursive. These member types are used when adding or removing members of a group as described in “The `nisgrpadm` Command” on page 177.

Member Types

- *Explicit.* An individual principal. Identified by principal name. The name does not have to be fully qualified if entered from its default domain.
- *Implicit.* All the NIS+ principals who belong to an NIS+ domain. They are identified by their domain name, preceded by the `*` symbol and a dot. The operation you select applies to all the members in the group.
- *Recursive.* All the NIS+ principals that are members of another NIS+ group. They are identified by their NIS+ group name, preceded by the `@` symbol. The operation you select applies to all the members in the group.

Nonmember Types

NIS+ groups also accept nonmembers in all three categories: explicit, implicit, and recursive. Nonmembers are principals specifically excluded from a group that they otherwise would be part of.

Nonmembers are identified by a minus sign in front of their name:

- *Explicit-nonmember*. Identified by a minus sign in front of the principal name.
- *Implicit nonmember*. Identified by a minus sign, * symbol, and dot in front of the domain name.
- *Recursive nonmember*. Identified by a minus sign and @ symbol in front of the group name.

Using Member Types

Note – The order in which inclusions and exclusions are entered does not matter. Exclusions always take precedence over inclusions. Thus, if a principal is a member of an included implicit domain and *also* a member of an excluded recursive group, then that principal is not included.

Thus, when using the `nisgrpadm` command, you can specify group members and nonmembers as shown in Table 9-2:

Table 9-2 Specifying Group Members and Nonmembers

Type of member	Syntax
Explicit member	<code>username.domain</code>
Implicit member	<code>*.domain</code>
Recursive member	<code>@groupname.domain</code>
Explicit nonmember	<code>-username.domain</code>
Implicit nonmember	<code>-* .domain</code>
Recursive nonmember	<code>-@groupname.domain</code>

Using niscat With Groups

The `niscat -o` command can be used to list the object properties and membership of an NIS+ group.

Listing the Object Properties of a Group

To list the object properties of a group, you must have read access to the `groups_dir` directory in which the group is stored. Use `niscat -o` and the group's fully qualified name, which must include its `groups_dir` subdirectory:

```
niscat -o group-name.groups_dir.domain-name
```

For example:

```
rootmaster# niscat -o sales.groups_dir.wiz.com.  
Object Name   : sales  
Owner         : rootmaster.wiz.com.  
Group         : sales.wiz.com.  
Domain        : groups_dir.wiz.com.  
Access Rights : ----rmcdr---r---  
Time to Live  : 1:0:0  
Object Type   : GROUP  
Group Flags   :  
Group Members : rootmaster.wiz.com.  
               topadmin.wiz.com.  
               @.admin.wiz.com.  
               *.sales.wiz.com.
```

A better arranged list of members is provided by the `nisgrpadm -l` command, on page 180.

Several of the group's properties are inherited from the `NIS_DEFAULTS` environment variable, unless they were overridden when the group was created. The group flags field is currently unused. In the list of group members, the `*` symbol identifies member domains and the `@` symbol identifies member groups.

The nisgrpadm Command

The `nisgrpadm` command creates, deletes, and performs miscellaneous administration operations on NIS+ groups. To use `nisgrpadm`, you must have access rights appropriate for the operation,

Table 9-3 Rights Required for `nisgrpadm` Command

This Operation	Requires This Access Right	To This Object
Create a group	Create	groups_dir directory
Destroy a group	Destroy	groups_dir directory
List the Members	Read	the group object
Add Members	Modify	the group object
Remove Members	Modify	the group object

The `nisgrpadm` has two main forms, one for working with groups and one for working with group members.

To create or delete a group, or to lists its members use this form:

```
nisgrpadm -c group-name.domain-name
nisgrpadm -d group-name
nisgrpadm -l group-name
```

To add or remove members, or determine if they belong to the group use this form (where *member...* can be any combination of the six membership types listed in Table 9-2 on page 175):

```
nisgrpadm -a group-name member...
nisgrpadm -r group-name member...
nisgrpadm -t group-name member...
```

All operations except create (`-c`) accept a partially qualified *group-names*. However, even for the `-c` option, `nisgrpadm` does not require the use of `groups_dir` in the *group-name* argument. In fact, it won't accept it.

Creating an NIS+ Group

To create an NIS+ group, you must have create rights to the `groups_dir` directory of the group's domain. Use the `-c` option and a fully qualified group name:

```
nisgrpadm -c group-name.domain-name
```

A newly created group contains no members. See “Adding Members to an NIS+ Group” on page 179 for information on how to specify who belongs to a group.

The example below creates three groups named `admin`. The first is in the `Wiz.com.` domain, the second in `sales.wiz.com.`, and the third in `manf.wiz.com.` All three are created on the master server of their respective domains.

```
rootmaster# nisgrpadm -c admin.wiz.com.
Group admin.wiz.com. created.
salesmaster# nisgrpadm -c admin.sales.wiz.com.
Group admin.sales.wiz.com. created.
engmaster# nisgrpadm -c admin.manf.wiz.com.
Group admin.manf.wiz.com. created.
```

The group you create will inherit all the object properties specified in the `NIS_DEFAULTS` variable; that is, its owner, owning group, access rights, time-to-live, and search path. You can view these defaults by using the `nisdefaults` command (described in Chapter 7, “Administering NIS+ Access Rights”). Used without options, it provides this output:

```
rootmaster# nisdefaults
Principal Name : rootmaster.wiz.com.
Domain Name   : Wiz.com.
Host Name     : rootmaster.Wiz.com.
Group Name    :
Access Rights : ----rmcdr---r---
Time to live  : 12:0:0
Search Path   : Wiz.com.
```

The owner is listed in the Principal Name: field. The owning group is listed only if you have set the `NIS_GROUP` environment variable.

Of course, you can override any of these defaults at the time you create the group by using the `-D` option:

```
salesmaster# nisgrpadm -D group=special.sales.wiz.com.-c admin.sales.wiz.com.  
Group admin.sales.wiz.com. created.
```

Deleting an NIS+ Group

To delete an NIS+ group, you must have destroy rights to the `groups_dir` directory in the group's domain. Use the `-d` option:

```
nisgrpadm -d group-name
```

If the default domain is set properly, you don't have to fully-qualify the group name. However, you should check first (use `nisdefaults`), because you could unintentionally delete a group in another domain. The example below deletes the `test.sales.wiz.com.` group.

```
salesmaster% nisgrpadm -d test.sales.wiz.com.  
Group `test.sales.wiz.com.` destroyed.
```

Adding Members to an NIS+ Group

To add members to an NIS+ group you must have modify rights to the group object. Use the `-a` option:

```
nisgrpadm -a group-name members...
```

As described in “Specifying Group Members” on page 174, you can add principals (explicit members), domains (implicit members), and groups (recursive members). You don't have to fully qualify the name of the group or the name of the members who belong to the default domain. This example

adds the NIS+ principals `panza` and `valjean`, both from the default domain, `sales.wiz.com.`, and the principal `makeba`, from the `manf.wiz.com.` domain, to the group `Ateam.sales.wiz.com`.

```
client% nisgrpadm -a Ateam panza valjean makeba.manf.wiz.com.  
Added panza.sales.wiz.com to group Ateam.sales.wiz.com  
Added valjean.sales.wiz.com to group Ateam.sales.wiz.com  
Added makeba.manf.wiz.com to group Ateam.sales.wiz.com
```

To verify the operation, use the `nisgrpadm -l` option. Look for the members under the `Explicit members` heading.

This example adds all the NIS+ principals in the `Wiz.com.` domain to the `Staff.wiz.com.` group. It is entered from a client in the `wiz.com.` domain. Note the `*` symbol and the dot in front of the domain name.

```
client% nisgrpadm -a Staff *.wiz.com.  
Added *.wiz.com. to group Staff.manf.wiz.com.
```

This example adds the NIS+ group `admin.wiz.com.` to the `admin.manf.wiz.com.` group. It is entered from a client of the `manf.wiz.com.` domain. Note the `@` symbol in front of the group name.

```
client% nisgrpadm -a admin @admin.wiz.com.  
Added @admin.wiz.com. to group admin.manf.wiz.com.
```

Listing the Members of an NIS+ Group

To list the members of an NIS+ group, you must have read rights to the group object. Use the `-l` option:

```
nisgrpadm -l group-name
```

This example lists the members of the `admin.manf.wiz.com.` group. It is entered from a client in the `manf.wiz.com.` group:

```
client% nisgrpadm -l admin
Group entry for admin.manf.wiz.com. group:
  No explicit members
  No implicit members:
  Recursive members:
    @admin.wiz.com.
  No explicit nonmembers
  No implicit nonmembers
  No recursive nonmembers
```

Removing Members From an NIS+ Group

To remove members from an NIS+ group, you must have modify rights to the group object. Use the `-r` option:

```
nisgrpadm -r group-name members...
```

This example removes the NIS+ principals `allende` and `hugo.manf.wiz.com.` from the `Ateam.sales.wiz.com` group. It is entered from a client in the `sales.wiz.com.` domain:

```
client% nisgrpadm -r Ateam allende hugo.manf.wiz.com.
Removed allende.sales.wiz.com. from group Ateam.sales.wiz.com.
Removed hugo.manf.wiz.com. from group Ateam.sales.wiz.com.
```

This example removes the `admin.wiz.com.` group from the `admin.manf.wiz.com.` group. It is entered from a client in the `manf.wiz.com.` domain:

```
client% nisgrpadm -r admin @admin.wiz.com.
Removed @admin.wiz.com. from group admin.manf.wiz.com.
```

Testing for Membership in an NIS+ Group

To find out whether an NIS+ principal is a member of a particular NIS+ group you must have read access to the group object. Use the `-t` option:

```
nisgrpadm -t group-name members...
```

This example tests whether the NIS+ principal `topadmin` belongs to the `admin.wiz.com.` group. It is entered from a client in the `Wiz.com.` domain.

```
client% nisgrpadm -t admin topadmin  
topadmin.wiz.com. is a member of group admin.wiz.com.
```

This example tests whether the NIS+ principal `jo`, from the `sales.wiz.com.` domain, belongs to the `admin.sales.wiz.com.` group. It is entered from a client in the `wiz.com.` domain.

```
client% nisgrpadm -t admin.sales.wiz.com. jo.sales.wiz.com.  
jo.sales.wiz.com. is a member of group admin.sales.wiz.com.
```


Administering NIS+ Directories

This chapter describes how to use the NIS+ directory administration commands to perform the following tasks:

<i>Listing the Object Properties of a Directory</i>	<i>page 184</i>
<i>Listing the Contents of a Directory—Terse</i>	<i>page 185</i>
<i>Listing the Contents of a Directory—Verbose</i>	<i>page 186</i>
<i>Creating a Directory</i>	<i>page 187</i>
<i>Adding a Replica to an Existing Directory</i>	<i>page 189</i>
<i>Removing a Directory</i>	<i>page 190</i>
<i>Disassociating a Replica From a Directory</i>	<i>page 191</i>
<i>Removing Nondirectory Objects</i>	<i>page 192</i>
<i>Starting a NIS-Compatible Daemon</i>	<i>page 193</i>
<i>Starting a DNS-Forwarding NIS-Compatible Daemon</i>	<i>page 194</i>
<i>Stopping the NIS+ Daemon</i>	<i>page 194</i>
<i>Initializing a Client</i>	<i>page 195</i>
<i>Initializing the Root Master Server</i>	<i>page 196</i>
<i>Starting the Cache Manager</i>	<i>page 197</i>
<i>Displaying the Contents of the NIS+ Cache</i>	<i>page 197</i>
<i>Displaying the Time of the Last Update</i>	<i>page 199</i>
<i>Pinging Replicas</i>	<i>page 199</i>

<i>Checkpointing a Directory</i>	<i>page 200</i>
<i>Changing the Time-to-Live of an Object</i>	<i>page 204</i>
<i>Changing the Time-to-Live of a Table Entry</i>	<i>page 205</i>

For a complete description of these commands and their syntax and options, see the NIS+ man pages.

Using the niscat Command With Directories

The `niscat -o` command can be used to list the object properties of an NIS+ directory. To use it, you must have read access to the directory object itself.

Listing the Object Properties of a Directory

To list the object properties of a directory, use `niscat -o` and the directory's name:

```
niscat -o directory-name
```

For example:

```
rootmaster# niscat -o wiz.com.
Object Name   : Wiz
Owner        : rootmaster.wiz.com.
Group        :
Domain       : Com.
Access Rights : r---rmcdr---r---
Time to Live  : 24:0:0
Object Type   : DIRECTORY
.
.
.
```

The nisls Command

The `nisls` command lists the contents of an NIS+ directory. To use it, you must have read rights to the directory object.

To display in terse format, use:

```
nisls [-dgLmMR] directory-name
```

To display in verbose format, use:

```
nisls -l [-gm] [-dLMR] directory-name
```

Table 10-1 Options for the `nisls` Command

Option	Purpose
-d	Directory object. Instead of listing a directory's contents, treat it like another object.
-L	Links. If the directory name is actually a link, the command follows the link and displays information about the linked directory.
-M	Master. Get the information from the master server only. Although this provides the most up-to-date information, it may take longer if the master server is busy.
-R	Recursive. List directories recursively. That is, if a directory contains other directories, their contents are displayed as well.
-l	Long. Display information in long format. Long format displays an object's type, creation time, owner, and access rights.
-g	Group. When displaying information in long format, display the directory's group owner instead of its owner.
-m	Modification time. When displaying information in long format, display the directory's modification time instead of its creation time.

Listing the Contents of a Directory—Terse

To list the contents of a directory in the default short format, use one or more of the options listed below and a directory name. If you don't supply a directory name, NIS+ will use the default directory.

```
nisls [-dLMR]  
nisls [-dLMR] directory-name
```

For example, this instance of `nisls` is entered from the root master server of the root domain `wiz.com.`:

```
rootmaster% nisls
wiz.com.:
org_dir
groups_dir
```

Here is another example entered from the root master server:

```
rootmaster% nisls -R Sales.wiz.com.
Sales.wiz.com.:
org_dir
groups_dir

groups_dir.Sales.wiz.com.:
admin

org_dir.Sales.wiz.com.:
auto_master
auto_home
bootparams
cred
.
.
.
```

Listing the Contents of a Directory—Verbose

To list the contents of a directory in the verbose format, use the `-l` option and one or more of the options listed below. The `-g` and `-m` options modify the attributes that are displayed. If you don't supply a directory name, NIS+ will use the default directory.

```
nisls -l [-gm] [-dLMR]
nisls -l [-gm] [-dLMR] directory-name
```

Here is an example, entered from the master server of the root domain wiz.com.:

```
rootmaster% nisls -l
wiz.com.:
D r---rmcdr---r--- rootmaster.wiz.com. date org_dir
D r---rmcdr---r--- rootmaster.wiz.com. date groups_dir
```

The nismkdir Command

This section describes how to add a nonroot directory and its master server to an existing system using the `nismkdir` command. An easier way to do this is with the `nissserver` script, as described in *NIS+ and DNS Setup and Configuration Guide*.

The `nismkdir` command creates a nonroot NIS+ directory and associates it with a master server. (To create a root directory, use the `nisinit -r` command, described in “The `nisinit` Command” on page 194.) The `nismkdir` command can also be used to add a replica to an existing directory.

There are several prerequisites to creating an NIS+ directory, as well as several related tasks. For a complete description, see *NIS+ and DNS Setup and Configuration Guide*.

To create a directory, use:

```
nismkdir [-m master-server] directory-name
```

To add a replica to an existing directory, use:

```
nismkdir -s replica-server directory-name
nismkdir -s replica-server org_dir.directory-name
nismkdir -s replica-server groups_dir.directory-name
```

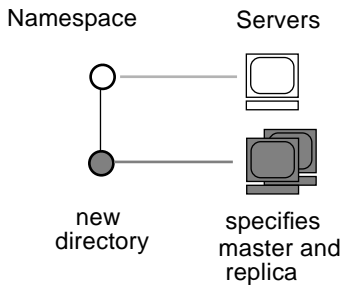
Creating a Directory

To create a directory, you must have create rights to its parent directory on the domain master server. First use the `-m` option to identify the master server and then the `-s` option to identify the replica, use:

```
nismkdir -m master directory
nismkdir -s replica directory
```



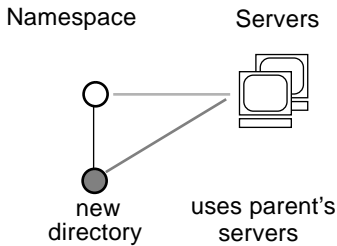
Caution – Always run `nismkdir` on the master server. Never run `nismkdir` on the replica machine. Running `nismkdir` on a replica creates communications problems between the master and the replica.



This example creates the `Sales.wiz.com.` directory and specifies its master server, `smaster.wiz.com.` and its replica, `repl.wiz.com.` It is entered from the root master server.

```

rootmaster% nismkdir -m smaster.wiz.com. Sales.wiz.com.
rootmaster% nismkdir -m smaster.wiz.com. org_dir.Sales.wiz.com.
rootmaster% nismkdir -m smaster.wiz.com.
groups_dir.Sales.wiz.com.
rootmaster% nismkdir -s repl.wiz.com. Sales.wiz.com.
rootmaster% nismkdir -s repl.wiz.com. org_dir.Sales.wiz.com.
rootmaster% nismkdir -s repl.wiz.com. groups_dir.Sales.wiz.com.
    
```

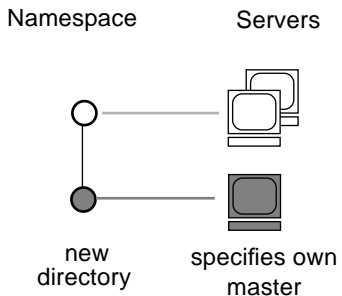


The `nismkdir` command allows you to use the parent directory's servers for the new directory instead of specifying its own. However, this should not be done except in the case of small networks. Here are two examples:

- The first example creates the `Sales.wiz.com.` directory and associates it with its parent directory's master and replica servers.

```

rootmaster% nismkdir Sales.wiz.com
    
```



- The second example creates the `Sales.wiz.com.` directory and specifies its own master server, `smaster.wiz.com.`

```

rootmaster% nismkdir -m smaster.wiz.com. Sales.wiz.com.
    
```

Since no replica server is specified, the new directory will have only a master server until you use `nismkdir` again to assign it a replica. If the `Sales.wiz.com.` domain already existed, the `nismkdir` command as shown above would have made `salesmaster.wiz.com.` its new master server and would have relegated its old master server to a replica.

Adding a Replica to an Existing Directory

This section describes how to add a replica server to an existing system using the `nismkdir` command. An easier way to do this is with the `nissserver` script as described in *NIS+ and DNS Setup and Configuration Guide*.

To assign a new replica server to an existing directory, use `nismkdir` on the master server with the `-s` option and the name of the existing directory, `org_dir`, and `groups_dir`:

```
nismkdir -s replica-server existing-directory-name
nismkdir -s replica-server org_dir.existing-directory-name
nismkdir -s replica-server groups_dir.existing-directory-name
```

The `nismkdir` command realizes that the directory already exists, so it does not recreate it. It only assigns it the additional replica. Here is an example with `repl` being the name of the new replica machine:

```
rootmaster% nismkdir -s repl.wiz.com. wiz.com.
rootmaster% nismkdir -s repl.wiz.com. org_dir.wiz.com.
rootmaster% nismkdir -s repl.wiz.com. groups_dir.wiz.com.
```

Note that you cannot assign a server to support its parent domain—unless, of course, it belongs to the root domain.



Caution – Always run `nismkdir` on the master server. Never run `nismkdir` on the replica machine. Running `nismkdir` on a replica creates communications problems between the master and the replica.

After running the three iterations of `nismkdir` as shown above, you need to run `nisping` from the master server on the three directories:

```
rootmaster# nisping wiz.com.  
rootmaster# nisping org_dir.wiz.com.  
rootmaster# nisping group_dir.wiz.com.
```

You should see results similar to these:

```
rootmaster# nisping wiz.com.  
Pinging replicas serving directory wiz.com. :  
Master server is rootmaster.wiz.com.  
    Last update occurred at Wed Nov 18 19:54:38 1995  
  
Replica server is repl.wiz.com.  
    Last update seen was Wed Nov 18 11:24:32 1995  
  
Pinging ... repl.wiz.com.
```

It is good practice to include `nisping` commands for each of these three directories in the master server's `cron` file so that each directory is “pinged” at least once every 24 hours after being updated.

The nisrmdir Command

The `nisrmdir` command can remove a directory or simply dissociate a replica server from a directory. When it removes a directory, NIS+ first disassociates the master and replica servers from the directory, and then removes the directory. To remove the directory, you must have `destroy` rights to its parent directory. To dissociate a replica server from a directory, you must have `modify` rights to the directory.

Removing a Directory

To remove an entire directory and dissociate its master and replica servers, use the `nisrmdir` command without any options:

```
nisrmdir directory-name
```


This example removes the `manf.wiz.com.` directory from beneath the `wiz.com.` directory:

```
rootmaster% nisrmdir manf.wiz.com.
```

Disassociating a Replica From a Directory

To dissociate a replica server from a directory, use the `nisrmdir` command with the `-s` option:

```
nisrmdir -s servername directory
```

This example disassociates the `manfreplica1` server from the `manf.wiz.com.` directory:

```
rootmaster% nisrmdir -s manfreplica1 manf.wiz.com.
```

The nisrm Command

The `nisrm` command is similar to the standard `rm` system command. It removes any NIS+ object from the namespace, except directories and nonempty tables. To use the `nisrm` command, you must have destroy rights to the object. However, if you don't, you can use the `-f` option, which tries to force the operation in spite of permissions.

You can remove group objects with the `nisgrpadm -d` command (see “Deleting an NIS+ Group” on page 179), and you can empty tables with `nistbladm -r` or `nistbladm -R` (see “Deleting a Table” on page 211).

To remove a nondirectory object, use:

```
nisrm [-if] object-name
```

Table 10-2 nisrm Syntax Options

Option	Purpose
-i	Inquire. Asks for confirmation prior to removing an object. If the <i>object-name</i> you provide is not fully qualified, this option is used automatically.
-f	Force. Attempts to force a removal even if you don't have the proper permissions. It attempts to change the permission by using the <code>nischmod</code> command, and then tries to remove the object again.

Removing Nondirectory Objects

To remove nondirectory objects, use the `nisrm` command and provide the object names:

```
nisrm object-name...
```

This example removes a group and a table from the namespace:

```
rootmaster% nisrm -i admins.wiz.com. groups.org_dir.wiz.com.  
Remove admins.wiz.com.? y  
Remove groups.org_dir.wiz.com.? y
```

The `rpc.nisd` Command

The `rpc.nisd` command starts the NIS+ daemon. The daemon can run in NIS-compatibility mode, which enables it to answer requests from NIS clients as well. You don't need any access rights to start the NIS+ daemon, but you should be aware of all its prerequisites and related tasks. They are described in *NIS+ and DNS Setup and Configuration Guide*.

By default, the NIS+ daemon starts with security level 2.

To start the daemon, use:

```
rpc.nisd
```

To start the daemon in NIS-compatibility mode, use:

```
rpc.nisd -Y [-B]
```

To start an NIS-compatible daemon with DNS forwarding capabilities, use:

```
rpc.nisd -Y -B
```

Table 10-3 Other `rpc.nisd` Syntax Options

Option	Purpose
<code>-S security-level</code>	Specifies a security level, where 0 means no NIS+ security and 2 provides full NIS+ security. (Level 1 is not supported.)
<code>-F</code>	Forces a checkpoint of the directory served by the daemon. This has the side effect of emptying the directory's transaction log and freeing disk space.

Starting the NIS+ Daemon

To start the NIS+ daemon on any server, use the command without options:

```
rpc.nisd
```

The daemon starts with security level 2, which is the default.

To start the daemon with security level 0, use the `-s` flag:

```
rpc.nisd -S 0
```

Starting a NIS-Compatible Daemon

You can start the NIS+ daemon in NIS-compatibility mode in any server, including the root master. Use the `-Y` (uppercase) option:

```
rpc.nisd -Y
```

If the server is rebooted, the daemon will not restart in NIS-compatibility mode unless you also uncomment the line that contains `EMULYP=Y` in the server's `/etc/init.d/rpc` file.

Starting a DNS-Forwarding NIS-Compatible Daemon

You can add DNS forwarding capabilities to an NIS+ daemon running in NIS-compatibility mode by adding the `-B` option to `rpc.nisd`:

```
rpc.nisd -Y -B
```

If the server is rebooted, the daemon will not restart in DNS-forwarding NIS-compatibility mode unless you also uncomment the line that contains `EMULYP="-Y -B"` in the server's `/etc/init.d/rpc` file and change it to:

```
EMULYP="-Y -B"
```

Stopping the NIS+ Daemon

To stop the NIS+ daemon, whether it is running in normal or NIS-compatibility mode, kill it as you would any other daemon: first find its process ID, then kill it:

```
rootmaster# ps -e | grep rpc.nisd
root 1081    1  61  16:43:33  ?        0:01  rpc.nisd -S 0
root 1087  1004  11  16:44:09  pts/1    0:00  grep rpc.nisd
rootmaster# kill 1081
```

The nisinit Command

This section describes how to initialize a workstation client using the `nisinit` command. An easier way to do this is with the `nisclient` script as described in *NIS+ and DNS Setup and Configuration Guide*.

The `nisinit` command initializes a workstation to be an NIS+ client. As with the `rpc.nisd` command, you don't need any access rights to use the `nisinit` command, but you should be aware of its prerequisites and related tasks. These are described in *NIS+ and DNS Setup and Configuration Guide*.

To initialize a client, use:

```
nisinit -c -B
nisinit -c -H hostname
nisinit -c -C filename
```

To initialize a root master server, use:

```
nisinit -r
```

Initializing a Client

You can initialize a client in three different ways:

- By host name
- By broadcast
- By cold-start file

Each way has different prerequisites and associated tasks. For instance, before you can initialize a client by host name, the client's `/etc/hosts` file must list the host name you will use and `nsswitch.conf` file must have `files` as the first choice on the `hosts` line. Complete instructions for each method, including prerequisites and associated tasks, are provided in *NIS+ and DNS Setup and Configuration Guide*. Following is a summary of the steps that use the `nisinit` command.

To initialize a client by host name, use the `-c` and `-H` options, and include the name of the server from which the client will obtain its cold-start file:

```
nisinit -c -H hostname
```

To initialize a client by cold-start file, use the `-c` and `-C` options, and provide the name of the cold-start file:

```
nisinit -c -C filename
```

To initialize a client by broadcast, use the `-c` and `-B` options:

```
nisinit -c -B
```

Initializing the Root Master Server

To initialize the root master server, use the `nisinit -r` command:

```
nisinit -r
```

The `nis_cachemgr` Command

The `nis_cachemgr` command starts the NIS+ cache manager program, which should run on all NIS+ clients. The cache manager maintains a cache of location information about the NIS+ servers that support the most frequently used directories in the namespace, including transport addresses, authentication information, and a time-to-live value.

At start-up the cache manager obtains its initial information from the client's cold-start file, and downloads it into the `/var/nis/NIS_SHARED_DIRCACHE` file.

The cache manager makes requests as a client workstation. Make sure the client workstation has the proper credentials, or instead of improving performance, the cache manager will degrade it.

Starting the Cache Manager

To start the cache manager, enter the `nis_cachemgr` command (with or without the `-i` option):

```
client% nis_cachemgr
client% nis_cachemgr -i
```

Without the `-i` option, the cache manager is restarted but it retains the information in the `/var/nis/NIS_SHARED_DIRCACHE` file. The information in the cold-start file is simply appended to the existing information in the file. The `-i` option clears the cache file and re-initializes it from the contents of the client's cold-start file.

To stop the cache manager, kill it as you would any other process.

The nisshowcache Command

The `nisshowcache` command displays the contents of a client's directory cache.

Displaying the Contents of the NIS+ Cache

The `nisshowcache` command is located in `/usr/lib/nis`. It displays only the cache header and the directory names. Here is an example entered from the root master server:

```
rootmaster# /usr/lib/nis/nisshowcache -v

Cold Start directory:
Name : wiz.com.
Type : NIS
Master Server :
    Name      : rootmaster.wiz.com.
    Public Key : Diffie-Hellman (192 bits)
    Universal addresses (3)
    .
    .
    .
Replicate:
    Name      : rootreplica1.wiz.com.
    Public Key : Diffie-Hellman (192 bits)
    Universal addresses (3)
    .
    .
    .
Time to live : 12:0:0
Default Access Rights :
```

The nisping Command

The `nisping` command pings replica servers, telling them to ask the master server for updates immediately. (The replicas normally wait a couple of minutes before executing this request.) Before pinging, the command checks the time of the last update received by each replica. If it is the same as the last update sent by the master, it does not ping the replica.

The `nisping` command can also checkpoint a directory. This consists of telling each server in the directory, including the master, to update its information on disk from the domain's transaction log.

To display the time of the last update, use:

```
/usr/lib/nis/nisping -u [domain]
```


To ping replicas, use:

```
/usr/lib/nis/nisping [domain]
/usr/lib/nis/nisping -H hostname [domain]
```

To checkpoint a directory, use:

```
/usr/lib/nis/nisping -C hostname [domain]
```

Displaying the Time of the Last Update

Use the `-u` option. It displays the update times for the master and replicas of the local domain, unless you specify a different domain name. It does not perform a ping.

```
/usr/lib/nis/nisping -u [domain]
```

Here is an example:

```
rootmaster# /usr/lib/nisping -u org_dir
Last updates for directory wiz.com.:
Master server is rootmaster.wiz.com.
    Last update occurred at Wed Nov 25 10:53:37 1992

Replica server is rootreplica1.wiz.com.
    Last update seen was Wed Nov 25 10:53:37 1992
```

Pinging Replicas

You can ping all the replicas in a domain, or one in particular. To ping all the replicas, use the command without options:

```
/usr/lib/nis/nisping
```

To ping all the replicas in a domain other than the local domain, append a domain name:

```
/usr/lib/nis/nisping domainname
```

Here is an example that pings all the replicas of the local domain, wiz.com.:

```
rootmaster# /usr/lib/nis/nisping org_dir
Pinging replicas serving directory wiz.com.:
Master server is rootmaster.wiz.com.
    Last update occurred at Wed Nov 25 10:53:37 1992

Replica server is rootreplica1.wiz.com.
    Last update seen was Wed Nov 18 11:24:32 1992

Pinging ... rootreplica1.wiz.com.
```

Since the update times were different, it proceeds with the ping. If the times had been identical, it would not have sent a ping.

You can also ping all the tables in all the directories on a single specified host. To ping all the tables in all the directories of a particular host, use the `-a` option:

```
/usr/lib/nis/nisping -a hostname
```

Checkpointing a Directory

To checkpoint a directory, use the `-c` option:

```
/usr/lib/nis/nisping -C directory-name
```

All the servers that support a domain, including the master, transfer their information from their `.log` files to disk. This erases the log files and frees disk space. While a server is checkpointing, it will still answer requests for service, but it will be unavailable for updates.

Here is an example of nisping output:

```

rootmaster# /usr/lib/nis/nisping -C
Checkpointing replicas serving directory wiz.com. :
Master server is rootmaster.wiz.com.
    Last update occurred at Wed May 25 10:53:37 1995

Master server is rootmaster.wiz.com.
checkpoint has been scheduled with rootmaster.wiz.com.
Replica server is rootreplica1.wiz.com.
    Last update seen was Wed May 25 10:53:37 1995

Replica server is rootreplica1.wiz.com.
checkpoint has been scheduled with rootmaster.wiz.com.

```

The nislog Command

The nislog command displays the contents of the transaction log.

```

/usr/sbin/nislog
/usr/sbin/nislog -h [number]
/usr/sbin/nislog -t [number]

```

Table 10-4 Options for the nislog Command

Option	Purpose
-h [num]	Display transactions starting with the head (beginning) of the log. If the number is omitted, the display begins with the first transaction. If the number 0 is entered, only the log header is displayed
-t [num]	Display transactions starting backward from the end (tail) of the log. If the number is omitted, the display begins with the last transaction. If the number 0 is entered, only the log header is displayed
-v	Verbose mode

Displaying the Contents of the Transaction Log

Each transaction consists of two parts: the particulars of the transaction and a copy of an object definition.

Here is an example that shows the transaction log entry that was made when the wiz.com. directory was first created. "XID" refers to the transaction ID.

```
rootmaster# /usr/sbin/nislog -h 1
NIS Log printing facility.
NIS Log dump:
    Log state : STABLE
Number of updates      : 48
Current XID            : 39
Size of log in bytes  : 18432
***UPDATES***
@@@@@@@@@@@@@@@@TRANSACTION@@@@@@@@@@@@@@@@
#00000, XID : 1
Time                  : Wed Nov 25 10:50:59 1992

Directory           : wiz.com.
Entry type          : ADD Name
Entry timestamp     : Wed Nov 25 10:50:59 1992
Principal           : rootmaster.wiz.com.
Object name         : org_dir.wiz.com.
.....Object.....
Object Name         : org_dir
Owner               : rootmaster.wiz.com.
Group               : admin.wiz.com.
Domain              : wiz.com.
Access Rights       : r---rmcdr---r---
Time to Live        : 24:0:0
Object Type         : DIRECTORY
Name : `org_dir.wiz.com.`
Type: NIS
Master Server       : rootmaster.wiz.com.
.
.
.....
@@@@@@@@@@@@@@@@TRANSACTION@@@@@@@@@@@@@@@@
#00000, XID : 2
```

The `nischttl` Command

The `nischttl` command changes the time-to-live value of objects or entries in the namespace. This time-to-live value is used by the cache manager to determine when to expire a cache entry. You can specify the time-to-live in total number of seconds or in a combination of days, hours, minutes, and seconds.

The time-to-live values you assign objects or entries should depend on the stability of the object. If an object is prone to frequent change, give it a low time-to-live value. If it is steady, give it a high one. A high time-to-live is a week; a low one is less than a minute. Password entries should have time-to-live values of about 12 hours to accommodate one password change per day. Entries in tables that don't change much, such as those in the RPC table, can have values of several weeks.

To change the time-to-live of an object, you must have modify rights to that object. To change the time-to-live of a table entry, you must have modify rights to the table, entry, or columns you wish to modify.

To display the current time-to-live value of an object or table entry, use the `nisdefaults -t` command, described in Chapter 7, "Administering NIS+ Access Rights."

To change the time-to-live value of objects, use:

```
nischttl time-to-live object-name
nischttl [-L] time-to-live object-name
```

To change the time-to-live value of entries, use:

```
nischttl time-to-live [column=value, ...], table-name
nischttl [-ALP] time-to-live [column=value, ...], table-name
```

Where *time-to-live* is expressed as:

- *Number of seconds.* A number with no letter is interpreted as a number of seconds. Thus, 1234 for TTL would be interpreted as 1234 seconds. A number followed by the letter *s* is also interpreted as a number of seconds.

Thus, `987s` for TTL would be interpreted as 987 seconds. When seconds are specified in combination with days, hours, or minutes, you must use the letter `s` to identify the seconds value.

- *Number of minutes.* A number followed by the letter `m` is interpreted as a number of minutes. Thus, `90m` for TTL would be interpreted as 90 minutes.
- *Number of hours.* A number followed by the letter `h` is interpreted as a number of hours. Thus, `9h` for TTL would be interpreted as 9 hours.
- *Number of days.* A number followed by the letter `d` is interpreted as a number of days. Thus, `7d` for TTL would be interpreted as 7 days.

These values may be used in combination. For example, a TTL value of `4d3h2m1s` would specify a time to live of four days, three hours, two minutes, and one second.

The following flags may also be used with the `nischttl` command:

Table 10-5 `nischttl` Syntax Options

Option	Purpose
-A	All. Apply the change to all the entries that match the <i>column=value</i> specifications that you supply.
-L	Links. Follow links and apply the change to the linked object or entry rather than the link itself.
-P	Path. Follow the path until there is one entry that satisfies the condition.

Changing the Time-to-Live of an Object

To change the time-to-live of an object, type the `nischttl` command with the *time-to-live* value and the *object-name*. You can add the `-L` command to extend the change to linked objects.

```
nischttl -L time-to-live object-name
```

You can specify the *time-to-live* in seconds by typing the number of seconds. Or you can specify a combination of days, hours, minutes, and seconds by using the suffixes `s`, `m`, `h`, and `d` to indicate the number of seconds, minutes, days, and hours. For example:

TTL of 86400 seconds
TTL of 24 hours
TTL of 2 days, 1 hour,
1 minute, and 1
second

```
client% nischt1 86400 sales.wiz.com.  
client% nischt1 24h sales.wiz.com.  
client% nischt1 2dlhlmls sales.wiz.com.
```

The first two commands change the time-to-live of the sales.wiz.com. directory to 86,400 seconds, or 24 hours. The third command changes the time-to-live of all the entries in a hosts table to 2 days, 1 hour, 1 minute, and 1 second.

Changing the Time-to-Live of a Table Entry

To change the time-to-live of entries, use the indexed entry format. You can use any of the options, -A, -L, or -P.

```
nischt1 [-ALP] time-to-live [column=value, . . .], table-name
```

These examples are similar to those above, but they change the value of table entries instead of objects:

C shell users should use quotes to prevent the shell from interpreting the square bracket ([]) as a metacharacter.

```
client% nischt1 86400 '[uid=99],passwd.org_dir.wiz.com.'  
client% nischt1 24h '[uid=99],passwd.org_dir.wiz.com.'  
client% nischt1 2dlhlmls '[name=fred],hosts.org_dir.wiz.com'
```


Administering NIS+ Tables

This chapter describes how to use the NIS+ table administration commands to perform the following tasks:

<i>Creating a New Table</i>	<i>page 209</i>
<i>Deleting a Table</i>	<i>page 211</i>
<i>Adding an Entry to a Table</i>	<i>page 211</i>
<i>Modifying a Table Entry</i>	<i>page 213</i>
<i>Removing a Single Entry From a Table</i>	<i>page 214</i>
<i>Removing Multiple Entries From a Table</i>	<i>page 215</i>
<i>Displaying the Contents of a Table</i>	<i>page 217</i>
<i>Displaying the Object Properties of a Table or Entry</i>	<i>page 217</i>
<i>Searching the First Column</i>	<i>page 221</i>
<i>Searching a Particular Column</i>	<i>page 222</i>
<i>Searching Multiple Columns</i>	<i>page 222</i>
<i>Creating a Link</i>	<i>page 223</i>
<i>Expanding a Directory Into an NIS+ Domain</i>	<i>page 225</i>
<i>Expanding a Directory Into an NIS-Compatible Domain</i>	<i>page 225</i>
<i>Loading Information From a File</i>	<i>page 228</i>
<i>Loading Data From an NIS Map</i>	<i>page 229</i>
<i>Dumping the Contents of an NIS+ Table to a File</i>	<i>page 231</i>

For a complete description of these commands and their syntax and options, see the NIS+ man pages.

The `nistbladm` Command

Some NIS+ table-related tasks can be performed more easily with Solstice AdminSuite tools if you have them available.

The `nistbladm` command is the primary NIS+ table administration command. With it, you can create, modify, and delete NIS+ tables and entries. To create a table, its directory must already exist. To add entries to the table, the table and columns must already be defined.

To create a table, you must have create rights to the directory under which you will create it. To delete a table, you must have destroy rights to the directory. To modify the contents of a table, whether to add, change, or delete entries, you must have modify rights to the table or the entries.

Syntax

To create or delete a table, use:

```
nistbladm -c table-type columnspec... tablename
nistbladm -d tablename

columnspec ::= column=[CSI,rights]
```

To add, modify, or remove entries, use:

```
nistbladm -a
nistbladm -A entry
nistbladm -m new-entry old-entry
nistbladm -r
nistbladm -R entry
entry ::= column=value ... tablename [column=value,...],tablename
```

The *columnspec* syntax is explained under the task “Creating a New Table” on page 209. The *entry* syntax is explained under the task “Adding an Entry to a Table” on page 211.

Creating a New Table

An NIS+ table must have at least one column and at least one of its columns must be searchable. To create an NIS+ table, use the `nistbladm` command with the `-c` option:

```
nistbladm -c table-type columnspec... tablename
```

The *table-type* is simply a string that identifies the table as belonging to a class of tables. It can be any string you choose.

The *columnspec* argument specifies the name and characteristics of each column. Enter one *columnspec* for each column you want in your new table. Separate the columns with spaces:

```
columnspec columnspec ...
```

Each *columnspec* entry has two to four components in the format:

```
name=type,rights:
```

Table 11-1 Columnspec Components

Component	Description
<i>name</i>	Name of the column
=	An equal sign which is required.
<i>type</i>	[Optional] The type of column specified by the letters S, I or C (see Table 11-2 on page 210). This component is optional. If no <i>type</i> is specified, the column becomes the default type.
<i>rights</i>	[Optional] Access rights. These access rights are over and above those granted to the table as a whole or to specific entries. If no <i>access</i> is specified, the column's access rights are those granted to the table as a whole, or to the entry. The syntax for access rights is described in "Specifying Access Rights in Commands" on page 120."

A column can be one of the following types:

Table 11-2 Column Types

Type	Description
S	Searchable. The <code>nismatch</code> command can search through the column.
I	Case-insensitive. When <code>nismatch</code> searches through the column, it will ignore case.
C	Encrypted.

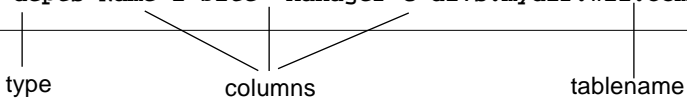
If you specify only access rights, you don't need to use a comma. If you include one or more of the S, I, or C flags, add a comma before the access rights.

This example creates a `depts`-type table named `divs` in the `mydir.wiz.com` directory (the `org_dir` directory is reserved for system tables).

```

master% nistbladm -c depts Name=I Site= Manager=C divs.mydir.wiz.com.

```



The table has three columns, `Name` (searchable, case-insensitive), `Site` (default characteristics), and `Manager` (Encrypted). Within any table there should be no two entries with the same values for all searchable columns.

In this example the same table is created with the addition of column-specific access rights applied to the first two columns:

```

master% nistbladm -c depts Name=I,w+m Site=w+m Manager=C divs.mydir.wiz.com.

```

For more information about specifying column access rights when creating a table, see “Setting Column Rights When Creating a Table” on page 130.

Deleting a Table

To delete a table, use the `-d` option and enter the table name:

```
nistbladm -d tablename
```

The table must be empty before you can delete it (see “Removing a Single Entry From a Table” on page 214). This example deletes the `divs` table from the `wiz.com.` directory:

```
rootmaster% nistbladm -d divs.wiz.com.
```

Adding an Entry to a Table

You can add an entry to a table in two ways:

- With the `-a` option. The `-a` option is recommended for administrators.
- With the `-A` option. The `-A` option is designed for applications.

Using the `-a` Option

The `-a` option adds an entry to a table unless the entry already exists, in which case it returns an error. To use it, you must specify a value for every column in the table:

```
nistbladm -a entry
```

To find the name of a particular column, use the `niscat -o` command. You can use two different syntaxes to specify an entry:

```
entry ::=column=value ... tablename [column=value, ...], tablename
```

The first consists of one or more *column=value* pairs, separated by spaces and followed by the table name. The second consists of one or more *column=value* pairs, separated by commas and enclosed in square brackets, followed by a comma and the table name. The second syntax is referred to as an *indexed-name*.

These two examples add the same entry to the `depts` table, but they each use a different form:

```
rootmaster% nistbladm -a Name='R&D' Site=SanFran Manager=vattel depts.wiz.com.
```

```
rootmaster% nistbladm -a [Name='R&D',Site=SanFran,Manager=vattel], depts.wiz.com.
```

Name	Site	Manager
R&D	SanFran	vattel

Note that in the two examples above, quotes are used to prevent the shell from interpreting the ampersand (&) in `R&D` as a metacharacter. C shell users should also use quotes to set off expressions using square brackets (`[]`).

You can only add one entry with each instance of the `nistbladm` command. This example adds three more entries to the `depts` table:

```
rootmaster% nistbladm -a [Name=Sales,Site=SanFran, Manager=tsosulu], depts.wiz.com.  
rootmaster% nistbladm -a [Name=Manf-1,Site=Emeryville, Manage=hosteen], depts.wiz.com.  
rootmaster% nistbladm -a [Name=Manf-2,Site=Sausalito, Manager=lincoln], depts.wiz.com.
```

Name	Site	Manager
R&D	SanFran	vattel
Sales	SanFran	tsosulu
Manf-1	Emeryville	hosteen
Manf-2	Sausalito	lincoln

Using the -A Option

The `-A` option is designed for applications. Like the `-a` option, it adds a new entry to a table. However, if the entry already exists, instead of exiting with an error, it changes the operation to modify, as if the `-m` option had been used instead. Unlike the `-m` option, however, with the `-A` option, you must specify all columns in the entry.

This example demonstrates how `-A` overwrites an existing entry:

Name	Site	Manager
R&D	SanFran	vattel
Sales	SanFran	tsosulu

```
rootmaster% nistbladm -A Name=Sales Site=SanFran Manager=jhill depts.wiz.com.
```

Name	Site	Manager
R&D	SanFran	vattel
Sales	SanFran	jhill

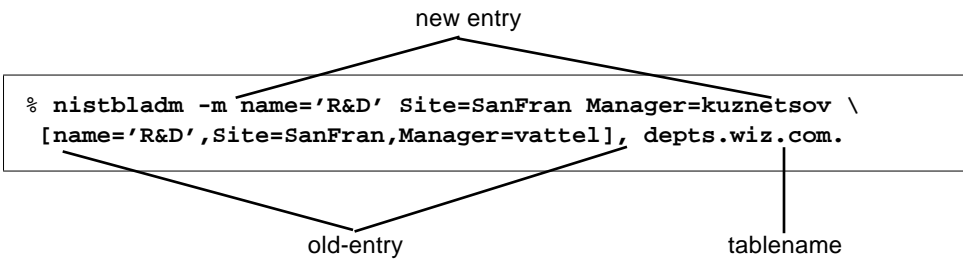
The `-a` option would have returned an error, since the entry specified by `Name=Sales Site=SanFran` already exists.

Modifying a Table Entry

To modify a table entry, use the `-m` option:

```
nistbladm -m new-entry old-entry
```

Specify the *new-entry* with a set of *column=value* pairs. Use an indexed name to specify the *old-entry* and the table name. This example modifies an entry in the `depts` table:



(Note that in the example above, quotes are used to prevent the shell from interpreting the ampersand (&) as a metacharacter. C shell users should also use quotes to set off expressions containing a bracket ([]) for the same reason.)

Name	Site	Manager
R&D	SanFran	kuznetsov
Sales	SanFran	jhill
Manf-1	Emeryville	hosteen
Manf-2	Sausalito	lincoln

Removing a Single Entry From a Table

To remove a single entry from a table, use the `-r` option:

```
nistbladm -r indexed-name
```

This example removes the `Manf-1` entry from the `depts` table:

```
rootmaster% nistbladm -r [Name=Manf-1,Site=Emeryville,Manager=hosteen],depts.wiz.com.
```


You can specify as few column values as you wish. If NIS+ finds duplicates, it does not remove any entry and returns an error message instead. Thus, you could have removed the `Manf-1` by specifying only the `Site` column value, as in this example:

```
rootmaster% nistbladm -r [Site=Emeryville],depts.wiz.com.
```

However, you could *not* have removed the `Sales` entry by specifying only the `Site` column value (`SanFran`), because two entries have that same value (`R&D` and `Sales`):

Name	Site	Manager
R&D	SanFran	kuznetsov
Sales	SanFran	jhill
Manf-1	Emeryville	hosteen
Manf-2	Sausalito	lincoln

Removing Multiple Entries From a Table

To remove multiple entries from a table, use the `-R` option:

```
nistbladm -R indexedname
```

As with the `-r` option, you can specify as few column values as you wish. Unlike the `-r` option, however, if NIS+ finds duplicates, it removes all of them. You can find the name of a table's column by using the `niscat -o` command. This example removes all entries in which the `Site` is `SanFran`:

```
rootmaster% nistbladm -R [Site=SanFran],depts.wiz.com.
```

Name	Site	Manager
Manf-1	Emeryville	hosteen
Manf-2	Sausalito	lincoln

You can use the `-R` option to remove all the entries from a table. Simply do not specify any column values, as in this example:

```
rootmaster% nistbladm -R [],depts.wiz.com.
```

The `niscat` Command

The `niscat` command displays the contents of an NIS+ table. However, you can also use it to display the object properties of the table. You must have read rights to the table, entries, or columns that you wish to display.

Syntax

To display the contents of a table, use:

```
niscat [-hM] tablename
```

To display the object properties of a table, use:

```
niscat -o tablename  
niscat -o entry
```

Table 11-3 `niscat` Syntax Options

Option	Description
-h	Header. Displays a header line above the table entries, listing the name of each column.
-M	Master. Displays only the entries of the table stored on the Master server. This ensures you get the most up-to-date information and should be used only for debugging.
-o	Object. Displays object information about the table, such as column names, properties, and servers.

Displaying the Contents of a Table

To display the contents of a table, use `niscat` with a table name:

```
niscat tablename
```

This example displays the contents of the table named `depts`.

```
rootmaster% niscat -h depts.wiz.com.  
#Name:Site:Manager  
R&D:SanFran:kuznetsov  
Sales:SanFran:jhill  
Manf-1:Emeryville:hosteen  
Manf-2:Sausalito:lincoln
```

Note – The symbol `*NP*` indicates that you do not have permission to view that entry. Permissions are granted on a table, column, or entry (row) basis. For more on access permissions, see Chapter 7, “Administering NIS+ Access Rights.”

Displaying the Object Properties of a Table or Entry

To list the object properties of a table use `niscat -o` and the table’s name:

```
niscat -o tablename.org_dir
```

To display the object properties of a table entry, use `niscat -o` and specify the entry with an indexed name:

```
entry ::=column=value ... tablename | [column=value, ...], tablename
```

Here are two examples, one for a table and one for a table entry:

```

rootmaster# niscat -o hosts.org_dir.wiz.com.
Object Name      : hosts
Owner            : rootmaster.wiz.com.
Group            : admin.wiz.com.
Domain           : org_dir.wiz.com.
Access Rights    : ----rmcdr---r---
Time to Live     : 12:0:0
Object Type      : TABLE
Table Type       : hosts_tbl
Number of Columns : 4
Character Separator :
Search Path      :
Columns          :
    [0]          Name           : cname
                  Attributes    : (SEARCHABLE, TEXTUAL DATA, CASE INS
                  Access Rights : -----
    [1]          Name           : name
                  Attributes    : (SEARCHABLE, TEXTUAL DATA, CASE INS
                  Access Rights : -----
    [2]          Name           : addr
                  Attributes    : (SEARCHABLE, TEXTUAL DATA, CASE INS
                  Access Rights : -----
    [3]          Name           : comment
                  Attributes    : (TEXTUAL DATA)
                  Access Rights : -----

```

```

rootmaster# niscat -o [name=rootmaster],hosts.org_dir.wiz.com.
Object Name      : hosts
Owner            : rootmaster.wiz.com.
Group            : admin.wiz.com.
Domain           : org_dir.wiz.com.
Access Rights    : ----rmcdr---r---
Time to Live     : 12:0:0
Object Type      : ENTRY
                  Entry data of type hosts_tbl
                  Entry has 4 columns.
                  .
                  .
                  .
#

```

The `nismatch` and `nisgrep` Commands

The `nismatch` and `nisgrep` commands search through NIS+ tables for entries that match a particular string or regular expression, respectively. They display either the entries themselves or a count of how many entries matched. The differences between the `nismatch` and `nisgrep` commands are highlighted in Table 11-4 below.

Table 11-4 Comparison of `nisgrep` and `nismatch`

Characteristics	<code>nismatch</code>	<code>nisgrep</code>
Search criteria	Accepts text only	Accepts regular expressions
Speed	Faster	Slower
Searches through	Searchable columns only	All columns, whether searchable or not
Syntax of search criteria	<code>column=string ... tablename [column=string, ...], t ablename</code>	<code>column=exp ... tablename</code>

The tasks and examples in this section describe the syntax for both commands.

To use either command, you must have read access to the table you are searching through.

The examples in this section are based on the values in the following table, named `depts.wiz.com`. Only the first two columns are searchable.

Table 11-5 `depts.wiz.com`. Example Table

Name (S)	Site (S)	Manager
R&D	SanFran	kuznetsov
Sales	SanFran	jhill
Manf-1	Emeryville	hosteen
Manf-2	Sausalito	lincoln
Shipping-1	Emeryville	tsosulu
Shipping-2	Sausalito	katabami
Service	Sparks	franklin

About Regular Expressions

Regular expressions are combinations of text and symbols that you can use to search for special configurations of column values. For example, the regular expression `'Hello'` searches for a value that begins with `Hello`. When using a regular expression in the command line, be sure to enclose it in quotes, since many of the regular expression symbols have special meaning to the Bourne and C shells. For example:

```
rootmaster% nisgrep -h greeting='Hello' phrases.wiz.com.
```

The regular expression symbols are summarized in Table 11-6, below.

Table 11-6 Regular Expression Symbols

Symbol	Description
<code>^string</code>	Find a value that begins with <i>string</i> .
<code>string\$</code>	Find a value that ends with <i>string</i> .
<code>.</code>	Find a value that has a number characters equal to the number of periods.
<code>[chars]</code>	Find a value that contains any of the characters in the brackets.
<code>*expr</code>	Find a value that has zero or more matches of the <i>expr</i> .
<code>+</code>	Find something that appears one or more times.
<code>?</code>	Find any value.
<code>\ 's-char'</code>	Find a special character, such as <code>?</code> or <code>\$</code> .
<code>x y</code>	Find a character that is either <i>x</i> or <i>y</i> .

Syntax

To search through the first column, use:

```
nismatch string tablename
nisgrep reg-exp tablename
```

To search through a particular column, use:

```
nismatch column=string tablename
nisgrep column=reg-exp tablename
```

To search through multiple columns, use:

```
nismatch column=string ... tablename
nismatch [column=string,...],tablename
nisgrep column=reg-exp ... tablename
```

Table 11-7 nismatch and nisgrep Syntax Options

Option	Description
-c	Count. Instead of the entries themselves, displays a count of the entries that matched the search criteria.
-h	Header. Displays a header line above the entries, listing the name of each column.
-M	Master. Displays only the entries of the table stored on the master server. This ensures you get the most up-to-date information and should be used only for debugging.

Searching the First Column

To search for a particular value in the first column of a table, simply enter the first column value and a *tablename*. In *nismatch*, the value must be a string. In *nisgrep*, the value must be a regular expression.

```
nismatch [-h] string tablename
nisgrep [-h] reg-expression tablename
```

This example searches through the *depts* table for all the entries whose first column has a value of *R&D*:

```
rootmaster% nismatch -h 'R&D' depts.wiz.com.
rootmaster% nisgrep -h 'R&D' depts.wiz.com.
```

Quotes are used in the *R&D* expression to prevent the shell from interpreting the ampersand (&) as a metacharacter.

Searching a Particular Column

To search through a particular column other than the first, use the following syntax:

```
nismatch column=string tablename
nisgrep column=reg-expression tablename
```

This example searches through the depts table for all the entries whose second column has a value of SanFran:

```
rootmaster% nismatch -h Site=SanFran depts.wiz.com
rootmaster% nisgrep -h Site=SanFran depts.wiz.com
```

Searching Multiple Columns

To search for entries with matches in two or more columns, use the following syntax:

```
nismatch [-h] column=string ... tablename nismatch [-h] [column=string,...],tablename
nisgrep [-h] column=reg-exp ... tablename
```

This example searches for entries whose second column has a value of SanFran and whose third column has a value of jhill:

```
rootmaster% nismatch -h [Site=SanFran,Manager=jhill],
depts.wiz.com.
rootmaster% nisgrep -h Site=SanFran Manager=jhill depts.wiz.com.
```

The nisl command

The `nisl` command creates symbolic links between NIS+ objects and table entries. You can use it to link objects to objects or objects to table entries. (You cannot create a link that originates with a table entry.) All NIS+ administration commands accept the `-L` flag, which directs them to follow links between NIS+ objects.

To create a link to another object or entry, you must have modify rights to the source object; that is, the one that will point to the other object or entry.



Caution – Never link a cred table. Each `org_dir` directory should have its own cred table. Do not use a link to some other `org_dir` cred table.

Syntax

To create a link, use:

```
nisl n source target
```

Table 11-8 nisl n Syntax Options

Option	Description
-L	Follow links. If the <i>source</i> is itself a link, the new link will not be linked to it, but to that link's original source.
-D	Defaults. Specify a different set of defaults for the linked object. Defaults are described in "Specifying Nondefault Security Values at Creation Time" on page 127.

Creating a Link

To create a link between objects, specify both object names: first the *source*, and then the *target*. To create links between objects and entries use indexed names.

```
nisl n source-object target-object
nisl n [column=value,...],tablename target-object
```

The `nissetup` Command

The `nisserver` script is easier to use for this purpose than the `nissetup` command. See *NIS+ and DNS Setup and Configuration Guide* for information on using the `nisserver` script.

The `nissetup` command expands an existing NIS+ directory object into a domain by creating the `org_dir` and `groups_dir` directories, and a full set of NIS+ tables. It does not, however, populate the tables with data. For that, you will need the `nisaddent` command, described in “The `nisaddent` Command” on page 226. Expanding a directory into a domain is part of the process of setting up a domain.

The `nissetup` command can expand a directory into a domain that supports NIS clients as well.

To use `nissetup`, you must have modify rights to the directory under which you’ll store the tables.

Syntax

To expand a directory into an NIS+ domain, use:

```
/usr/lib/nis/nissetup
/usr/lib/nis/nissetup directory-name
```

To expand a directory into an NIS-compatible NIS+ domain, use:

```
/usr/lib/nis/nissetup -Y
/usr/lib/nis/nissetup -Y directory-name
```

Expanding a Directory Into an NIS+ Domain

You can use the `nissetup` command with or without a directory name. If you don't supply the directory name, it uses the default directory. Each object that is added is listed in the output.

```
rootmaster# /usr/lib/nis/nissetup wiz.com.  
org_dir.wiz.com. created  
groups_dir.wiz.com. created  
auto_master.org_dir.wiz.com. created  
auto_home.org_dir.wiz.com. created  
bootparams.org_dir.wiz.com. created  
cred.org_dir.wiz.com. created  
ethers.org_dir.wiz.com. created  
group.org_dir.wiz.com. created  
hosts.org_dir.wiz.com. created  
mail_aliases.org_dir.wiz.com. created  
sendmailvars.org_dir.wiz.com. created  
netmasks.org_dir.wiz.com. created  
netgroup.org_dir.wiz.com. created  
networks.org_dir.wiz.com. created  
passwd.org_dir.wiz.com. created  
protocols.org_dir.wiz.com. created  
rpc.org_dir.wiz.com. created  
services.org_dir.wiz.com. created  
timezone.org_dir.wiz.com. created
```

Expanding a Directory Into an NIS-Compatible Domain

To expand a directory into a domain that supports NIS+ and NIS client requests, use the `-Y` flag. The tables are created with read rights for the nobody class so that NIS clients requests can access them.

```
rootmaster# /usr/lib/nis/nissetup -Y Test.wiz.com.
```

The `nisaddent` Command

The `nisaddent` command loads information from text files or NIS maps into NIS+ tables. It can also dump the contents of NIS+ tables back into text files. If you are populating NIS+ tables for the first time, see the instructions in *NIS+ and DNS Setup and Configuration Guide*. It describes all the prerequisites and related tasks.

You can use `nisaddent` to transfer information from one NIS+ table to another (for example, to the same type of table in another domain), but not directly. First, you need to dump the contents of the table into a file, and then load the file into the other table. Be sure, though, that the information in the file is formatted properly. Appendix C, “Information in NIS+ Tables,” describes the format required for each table.

When you load information into a table, you can use any of three options: replace, append, or merge. The append option simply adds the source entries to the NIS+ table. With the replace option, NIS+ first deletes all existing entries in the table and then adds the entries from the source. In a large table, this adds a large set of entries into the table’s `.log` file (one set for removing the existing entries, another for adding the new ones), taking up space in `/var/nis` and making propagation to replicas time consuming.

The merge option produces the same result as the replace option but uses a different process, one that can greatly reduce the number of operations that must be sent to the replicas. With the merge option, NIS+ handles three types of entries differently:

- Entries that exist only in the source are *added* to the table
- Entries that exist in both the source and the table are *updated* in the table
- Entries that exist only in the NIS+ table are *deleted* from the table

When updating a large table with a file or map whose contents are not greatly different from those of the table, the merge option can spare the server a great many operations. Because the merge option deletes only the entries that are not duplicated in the source (the replace option deletes *all* entries, indiscriminately), it saves one delete and one add operation for every duplicate entry.

If you are loading information into the tables for the first time, you must have create rights to the table object. If you are overwriting information in the tables, you must have modify rights to the tables.

Syntax

To load information from text files, use:

```

/usr/lib/nis/nisaddent -f filename table-type [domain]
/usr/lib/nis/nisaddent -f filename -t tablename table-type
[domain]

```

To load information from NIS maps, use:

```

/usr/lib/nis/nisaddent -y NISdomain table-type [domain]
/usr/lib/nis/nisaddent -y NISdomain -t tablename table-type [domain]
/usr/lib/nis/nisaddent -Y map table-type [domain]
/usr/lib/nis/nisaddent -Y map -t tablename table-type [domain]

```

To dump information from an NIS+ table to a file, use:

```

/usr/lib/nis/nisaddent -d [-t tablename] tabletype > filename

```

Table 11-9 nisaddent Syntax Options

Option	Description
-a	Append. Contents of the source are appended to contents of the table.
-r	Replace. Contents of the source replace contents of the table.
-m	Merge. Contents of the source are merged with contents of the table.
-d	Dump. Contents of the NIS+ table are dumped to <code>stdout</code> .
-v	Verbose. The command prints verbose status messages.
-P	Follow path. If the command was unable to find a table, follow the search paths specified in the environment variable <code>NIS_PATH</code> .
-A	All data. Apply the operation to all the tables in the search path.
-M	Master server. Use the tables only in the master server of the domain.
-D	Override defaults. For the new data being loaded into the tables, override existing defaults. For syntax, see “Specifying Nondefault Security Values at Creation Time” on page 127.

Loading Information From a File

You can transfer the contents of a file into an NIS+ table in several different ways:

- The `-f` option with no other option *replaces* the contents of *table-type* in the local domain with the contents of *filename*.

```
nisaddent -f filename table-type
```

- With the `-a` option, `-f` *appends* the contents of *filename* to *table-type*.

```
nisaddent -a -f filename table-type
```

- With the `-m` option, `-f` *merges* the contents of *filename* into the contents of *table-type*.

```
nisaddent -m -f filename table-type
```

The following two examples load the contents of a text file named `/etc/passwd.xfr` into the NIS+ `Passwd` table. The first is into a table in the local domain, the second into a table in another domain:

```
rootmaster# /usr/lib/nis/nisaddent -f /etc/passwd.xfr passwd
rootmaster# /usr/lib/nis/nisaddent -f /etc/shadow.xfr shadow

rootmaster# /usr/lib/nis/nisaddent -f /etc/passwd.xfr passwd sales.wiz.com.
rootmaster# /usr/lib/nis/nisaddent -f /etc/shadow.xfr shadow sales.wiz.com.
```

Note – When creating a NIS+ `passwd` table from `/etc` files, you must run `nisaddent` twice; once on the `/etc/passwd` file and once on the `/etc/shadow` file.

Another way is to use `stdin` as the source. However, you cannot use the `-m` option with `stdin`. You can use redirect (`>`) or pipe (`|`), but you cannot pipe into another domain.

Table 11-10 Using `cat` with `nisaddent`

Task	Command
Redirect	<code>cat filename > nisaddent table-type</code>
Redirect with append option	<code>cat filename > nisaddent -a table-type</code>
Redirect with append into another domain	<code>cat filename > nisaddent -a table-type NIS+ domain</code>
Pipe	<code>cat filename nisaddent table-type</code>
Pipe with append option	<code>cat filename nisaddent -a table-type</code>

If the NIS+ table is one of the automounter tables or a nonstandard table, add the `-t` option and the complete name of the NIS+ table.

```
master# nisaddent -f /etc/auto_home.xfr -t auto_home.org_dir.wiz.com. key-value
master# nisaddent -f /etc/auto_home.xfr -t auto_home.org_dir.wiz.com. key-value sales.wiz.com.
```

Loading Data From an NIS Map

`/var/yp/nisdomain` must be local files

You can transfer information from an NIS map in two different ways; either by specifying the NIS domain or by specifying the actual NIS map. If you specify the domain, NIS+ will figure out which map file in `/var/yp/nisdomain` to use as the source, based on the *table-type*.

Table 11-11 NIS+ Table Types and Matching NIS Map Names

NIS+ Table Type	NIS Map Name
Hosts	<code>hosts.byaddr</code>
Passwd	<code>passwd.byname</code>
Group	<code>group.byaddr</code>
Ethers	<code>ethers.byname</code>
Netmasks	<code>netmasks.byaddr</code>

Table 11-11 NIS+ Table Types and Matching NIS Map Names

NIS+ Table Type	NIS Map Name
Networks	networks.byname
Protocols	protocols.byname
RPC	rpc.bynumber
Services	services.byname

To transfer by specifying the NIS domain, use the `-y` (lowercase) option and provide the NIS domain in addition to the NIS+ table type.

Table replacement

```
nisaddent -y nisdomain table-type
```

Table append

```
nisaddent -a -y nisdomain table-type
```

Table merge

```
nisaddent -m -y nisdomain table-type
```

By default, `nisaddent` replaces the contents of the NIS+ table with the contents of the NIS map. Use the `-a` and `-m` options to append or merge. Here is an example that loads the NIS+ `passwd` table from its corresponding NIS map (`passwd.byname`) in the `old-wiz` domain:

```
rootmaster# /usr/lib/nis/nisaddent -y old-wiz passwd
```

This example does the same thing, but for the `sales.wiz.com.` domain instead of the local domain, `wiz.com.`

```
rootmaster# /usr/lib/nis/nisaddent -y old-wiz passwd sales.wiz.com.
```


If the NIS+ table is one of the automounter tables or a nonstandard table, add the `-t` option and the complete name of the NIS table, just as you would if the source were a file.

```
rootmaster# nisaddent -y old-wiz -t auto_home.org_dir.wiz.com. key-value
rootmaster# nisaddent -y old-wiz -t auto_home.org_dir.wiz.com. key-value sales.wiz.com.
```

If instead of using the map files for a domain, you prefer to specify a particular NIS map, use the `-Y` (uppercase) option and specify the map name.

```
rootmaster# nisaddent -Y hosts.byname hosts
rootmaster# nisaddent -Y hosts.byname hosts sales.wiz.com.
```

If the NIS map is one of the automounter maps or a non standard map, combine the `-Y` option with the `-t` option:

```
rootmaster# nisaddent -Y auto_home -t auto_home.org_dir.wiz.com. key-value
rootmaster# nisaddent -Y auto_home -t auto_home.org_dir.wiz.com. key-value sales.wiz.com.
```

Dumping the Contents of an NIS+ Table to a File

To dump the contents of an NIS+ table into a file, use the `-d` and `-t` options. The `-d` options tells the command to dump, and the `-t` option specifies the NIS+ table:

```
/usr/lib/nis/nisaddent -d [-t tablename] tabletype > filename
```


The Name Service Switch

The name service switch is often referred to as simply the **switch**

The name service switch is not really part of NIS+. However, it enables clients of `getXXbyYY()` routines, such as NIS+, to obtain their network information from one or more *sources* such as NIS+ tables, NIS maps, the DNS hosts table, or local `/etc` files. This chapter describes the switch, what it can do, and how it is used with NIS+.

<i>About the Name Service Switch</i>	<i>page 233</i>
<i>The nsswitch.conf Template Files</i>	<i>page 239</i>
<i>DNS Forwarding</i>	<i>page 241</i>
<i>Adding Compatibility With +/- Syntax</i>	<i>page 242</i>

About the Name Service Switch

An NIS+ client can obtain its information from one or more of the switch's sources in place of, or in addition, to NIS+ tables. For example, an NIS+ client could obtain its hosts information from an NIS+ table, its group information from NIS maps, and its password information from a local `/etc` file. In addition, it could specify the conditions under which the switch must use each source (see "Search Criteria" on page 235).

These choices are specified in a special configuration file called `nsswitch.conf`. This file is automatically loaded into every workstation's `/etc` directory by the Solaris 2.4 software, along with three alternate (template) versions:

- /etc/nsswitch.nisplus
- /etc/nsswitch.nis
- /etc/nsswitch.files

These alternate files contain the default switch configurations used by the NIS+ service, NIS, and local files. (See “The nsswitch.conf Template Files” on page 239.) No default file is provided for DNS, but you can edit any of these files to use DNS (see “DNS Forwarding” on page 241).

When the Solaris 2.4 software is first installed on a workstation, the installer must select the workstation’s default name service: NIS+, NIS, or local files. During the installation, the corresponding configuration file is copied into the /etc/nsswitch.conf file.

You can change the sources of information used by an NIS+ client by creating your own customized configuration file and copying it over /etc/nsswitch.conf. Its syntax is described below, and instructions are provided in *NIS+ and DNS Setup and Configuration Guide*.

Format of the nsswitch.conf File

The passwd entry in the nsswitch.conf file includes shadow information.

The nsswitch.conf file is essentially a list of 15 types of information and the sources that getXXbyYY() routines, such as NIS+, search for that information. The 15 types of information, not necessarily in this order, are:

aliases:	source(s)
bootparams:	source(s)
ethers:	source(s)
group:	source(s)
hosts:	source(s)
netgroup:	source(s)
netmasks:	source(s)
networks:	source(s)
passwd:	source(s)
protocols:	source(s)
publickey:	source
rpc:	source(s)
services:	source(s)
automount:	source(s)
sendmailvars	source(s)

Table 12-1 provides a description of *sources*.

Table 12-1 Possible Switch Sources

Source	Description
files	A local file stored in the client's <code>/etc</code> directory (for example, <code>/etc/passwd</code>)
nisplus	An NIS+ table
nis	An NIS map
compat	Only for the password and group entries, supports the old-style <code>+</code> or <code>-</code> syntax in the <code>/etc/passwd</code> , <code>/etc/shadow</code> , and <code>/etc/group</code> files. For both NIS and NIS+. You must use <code>passwd_compat: nisplus</code> for NIS+ (see “Adding Compatibility With <code>+/-</code> Syntax” on page 242).
dns	DNS, but only for the hosts entry

Search Criteria

If an information type has only one source, for example, `publickey: nisplus`, a routine using the switch searches for the information in that source only. If it finds the information, it returns a `success` status message; if it does not find the information, it stops searching and returns a different status message. What the routine does with the status message varies from routine to routine.

If a table has more than one source, the switch directs the routine to start searching for the information in the first source. If it finds the information, it returns a `success` status message; if it does not find the information there, it tries the next source. The routine will search through all of the sources until it has found the information it needs, it is halted by encountering a `return` condition, or it has tried all of the sources without success. If all of the listed sources are searched without finding the information, the routine stops searching and returns a `non-success` status message. What the routine does with the status message varies from routine to routine.

Switch Status Messages

If a routine finds the information it returns a `success` status message; if it does not find the information it is looking for, it returns one of three unsuccessful status messages, depending on the reason for not finding the information. The four possible status messages are listed in Table 12-2.

Table 12-2 Switch Status Messages

Status	Meaning
SUCCESS	The requested entry was found in the source (NIS+ table, NIS map, or <code>/etc</code> file).
UNAVAIL	The source is not responding or is unavailable. In other words, the NIS+ table, NIS map, or <code>/etc</code> file could not be found or accessed.
NOTFOUND	The source responded with “No such entry,” In other words, the table, map, or file was found, but it did not contain the needed information.
TRYAGAIN	The source is busy; it might respond next time. In other words, the table, map, or file was found, but it could not respond to the query.

Switch Action Options

You can instruct the switch to respond to status messages with either of these two *actions* shown in Table 12-3.

Table 12-3 Responses to Switch Status Messages

Action	Meaning
<code>return</code>	Stop looking for the information.
<code>continue</code>	Try the next source, if there is one.

Default Search Criteria

The combination of `nsswitch.conf` file status message and action option determine what the routine does at each step. This combination of status and action is called the search *criteria*.

If a routine searches through all of the sources listed in the switch file without finding what it is looking for, the routine returns with a NOTFOUND status.

The switch's default search criteria are the same for every source. Described in terms of the status messages listed above, they are:

- **SUCCESS=return.** Stop looking for the information and proceed using the information that has been found.
- **UNAVAIL=continue.** Go to the next `nsswitch.conf` file source and continue searching. If this is the last source, return with a NOTFOUND status.
- **NOTFOUND=continue.** Go to the next `nsswitch.conf` file source and continue searching. If this is the last source, return with a NOTFOUND status.
- **TRYAGAIN=continue.** Go to the next `nsswitch.conf` file source and continue searching. If this is the last source, return with a NOTFOUND status.

Because these are the default search criteria, they are assumed (not explicitly specified). You can change these default search criteria by explicitly specifying some other criteria using the `STATUS=action` syntax shown above. For example:

```
hosts:      nis
networks:  nis [NOTFOUND=return] files
```

- `hosts: nis`. This is an example of the default search criteria. Any routine looking for host information will search the NIS `hosts` map. If the `hosts` map is not available, does not contain the needed information, or is busy, the routine will return with either an UNAVAIL, NOTFOUND, or TRYAGAIN status message.
- `networks: nis [NOTFOUND=return] files`. This line specifies a nondefault criterion for the NOTFOUND status. (Nondefault criteria are delimited by square brackets.) In this case, a routine would return with a SUCCESS status if it found the information in the NIS `networks` map or continue on to search the `/etc` file if the NIS `networks` map was not found (UNAVAILABLE status) or was found but did not respond (TRYAGAIN status). However, if the NIS map was found and accessed, but did not contain the needed information, the routine would return with a NOTFOUND status *without* searching the `/etc` file.

What if the Syntax Is Wrong?

Client library routines contain compiled-in default entries that are used if an entry in the `nsswitch.conf` file is either missing or syntactically incorrect. These entries are the same as the default `nsswitch.conf` file.

The name service switch assumes that the spelling of table and source names is correct. If you misspell a table or source name, the switch uses the default values instead.

Auto_home and Auto_master

The information for the `auto_home` and `auto_master` tables is combined into one category, called “automount.”

Timezone

The `timezone` table does not use the switch, so it is not included in the list.

Comments in nsswitch.conf Files

Any `nsswitch.conf` file line beginning with a hash character (`#`) is interpreted as a comment line and are thus ignored by routines that search the file.

When a hash character (`#`) is included in the middle of the line, characters to the *left* of the hash mark (before the hash mark) are interpreted by routines that search the `nsswitch.conf` file; characters to the right of the hash mark (after the hash mark) are interpreted as comments and acted upon.

Table 12-4 Switch File Comment Examples

Type of Line	Example
Comment line (not interpreted).	<code># hosts: nisplus [NOTFOUND=return] files</code>
Fully interpreted line.	<code>hosts: nisplus [NOTFOUND=return] files</code>
Partially interpreted line (the <code>files</code> element not interpreted)	<code>hosts: nisplus [NOTFOUND=return] # files</code>

The `nsswitch.conf` *Template Files*

Three `nsswitch.conf` template files are provided with the Solaris 2.x release. Each of them provides a different default set of primary and subsequent information sources. The three template files are:

- *NIS+ template file.* The `nsswitch.nisplus` configuration file specifies NIS+ as the primary source for all information except `passwd`, `group`, `automount`, and `aliases`. For those four files, the primary source is local `/etc` files and the secondary source is an NIS+ table. The `[NOTFOUND=return]` search criterion instructs the switch to stop searching the NIS+ tables if it receives a “No such entry” message from them. It searches through local files only if the NIS+ server is unavailable. (See Code Example 12-1 on page 240.)
- *NIS template file.* The `nsswitch.nis` configuration file is almost identical to the NIS+ configuration file, except that it specifies NIS maps in place of NIS+ tables. Because the search order for `passwd` and `group` is `files nis`, you don’t need to place the `+` entry in the `/etc/passwd` and `/etc/group` files. (See Code Example 12-2 on page 240.)
- *Files template file.* The `nsswitch.files` configuration file specifies local `/etc` files as the only source of information for the workstation. There is no “files” source for `netgroup`, so the client simply won’t use it. (See Code Example 12-3 on page 241.)

Copy the template file that most closely meets your requirements to `nsswitch.conf` and then modify `nsswitch.conf` as needed. (See the switch chapter of *NIS+ and DNS Setup and Configuration Guide* for a detailed description of this process.)

For example, to use the NIS+ template file, you would type the following command:

```
mymachine# cp nsswitch.nisplus nsswitch.conf
```

Note – Note that the keyserver reads the `publickey` entry in the name service switch configuration file only when the keyserver is started. As a result, if you change the switch configuration file, the keyserver does not become aware of changes to the `publickey` entry until it is restarted.

Switch Template File Examples

Here are the three template files with all the comments stripped out:

Code Example 12-1 NIS+ nsswitch.conf Template File

```
passwd:      files nisplus
group:       files nisplus
hosts:       nisplus [NOTFOUND=return] files
services:   nisplus [NOTFOUND=return] files
networks:   nisplus [NOTFOUND=return] files
protocols:  nisplus [NOTFOUND=return] files
rpc:         nisplus [NOTFOUND=return] files
ethers:     nisplus [NOTFOUND=return] files
netmasks:  nisplus [NOTFOUND=return] files
bootparams: nisplus [NOTFOUND=return] files
publickey:  nisplus
netgroup:   nisplus
automount:  files nisplus
aliases:    files nisplus
```

Code Example 12-2 NIS nsswitch.conf Template File

```
passwd:      files nis
group:       files nis
hosts:       nis [NOTFOUND=return] files
services:   nis [NOTFOUND=return] files
networks:   nis [NOTFOUND=return] files
protocols:  nis [NOTFOUND=return] files
rpc:         nis [NOTFOUND=return] files
ethers:     nis [NOTFOUND=return] files
netmasks:  nis [NOTFOUND=return] files
bootparams: nis [NOTFOUND=return] files
publickey:  nis [NOTFOUND=return] files
netgroup:   nis
automount:  files nis
aliases:    files nis
```

Code Example 12-3 Files `nsswitch.conf` Template File

```
passwd:      files
group:       files
hosts:       files
networks:    files
protocols:   files
rpc:         files
ethers:      files
netmasks:   files
bootparams:  files
publickey:   files
netgroup:    files
automount:   files
aliases:     files
services:    files
```

Default `nsswitch.conf` File

The default `nsswitch.conf` file shipped with the Solaris 2.4 software is actually a copy of the `nsswitch.nis` file, described below. You can change it to the NIS+ version by copying the `nsswitch.nisplus` file over the `/etc/nsswitch.conf` file.

DNS Forwarding

The `nsswitch.conf` file also controls DNS forwarding for clients as described in the following subsections.

Note – The NIS+ client must have a properly configured `/etc/resolv.conf` file (as described in *NIS+ and DNS Setup and Configuration Guide*).

See the switch file chapter of *NIS+ and DNS Setup and Configuration Guide* for step-by-step instructions on enabling DNS forwarding for NIS+ and NIS clients.

DNS Forwarding for NIS+ Clients

NIS+ clients do *not* have implicit DNS forwarding capabilities like NIS clients do. Instead, they take advantage of the switch. To provide DNS forwarding capabilities to an NIS+ client, change its `hosts` entry to:

```
hosts: nisplus dns [NOTFOUND=return] files
```

DNS Forwarding for NIS Clients

If an NIS client is using the DNS forwarding capability of a NIS-compatible NIS+ server, its `nsswitch.conf` file should *not* have the following syntax for the `hosts` file:

```
hosts: nis dns files
```

Since DNS forwarding automatically forwards host requests to DNS, the syntax shown above would cause the NIS+ server to forward unsuccessful requests to the DNS servers twice, impacting performance. To take best advantage of DNS forwarding, use the default syntax for the `nsswitch.nis` file, as shown Code Example 12-2 on page 240.

Adding Compatibility With +/- Syntax

You can add to your `nsswitch.conf` file compatibility with the `+/-` syntax sometimes used in `/etc/passwd`, `/etc/shadow`, and `/etc/group` files.

- *NIS+*. To provide +/- semantics with NIS+, change the `passwd` and `groups` sources to `compat` and add a `passwd_compat: nisplus` entry to the `nsswitch.conf` file after the `passwd` or `group` entry as shown below:

```
passwd: compat
passwd_compat: nisplus
group: compat
group_compat: nisplus
```

This specifies that client routines obtain their network information from `/etc` files and NIS+ tables as indicated by the +/- entries in the files.

- *NIS*. To provides the same syntax as in the SunOS 4.1 release, change the `passwd` and `groups` sources to `compat`. This specifies that client routines obtain their network information from `/etc` files and NIS maps as indicated by the +/- entries in the files.

```
passwd: compat
group: compat
```

Note – Users working on a client machine being served by a NIS+ server running in compatibility mode cannot run `ypcat` on the `netgroup` table. Doing so will give you results as if the table were empty even if it has entries.

See the `switch` file chapter of *NIS+ and DNS Setup and Configuration Guide* for step by step instructions on adding +/- semantics to an `nsswitch.conf` file.

Removing NIS+

This chapter describes how to use the NIS+ directory administration commands to perform the following tasks:

<i>Removing NIS+ From a Client Machine</i>	<i>page 245</i>
<i>Removing NIS+ From a Server</i>	<i>page 246</i>
<i>Removing the NIS+ Namespace</i>	<i>page 248</i>

Removing NIS+ From a Client Machine

This section described how to remove NIS+ from a client machine. Keep in mind that removing NIS+ from a client machine does not remove the NIS+ name service from your network. See “Removing the NIS+ Namespace” on page 248 for information on removing the NIS+ name service from a network and returning to either NIS or `/etc` files for name purposes.

Removing NIS+ That Was Installed Using `nisclient`

To remove NIS+ from a client machine that was set up as an NIS+ client using the `nisclient -i` script as described in *NIS+ and DNS Setup and Configuration Guide*, simply run `nisclient` with the `-r` option:

```
clientmachine# nisclient -r
```

`nisclient -r` simply undoes the most recent iteration of `nisclient -i`; it restores the previous naming system used by the client, such as NIS or `/etc` files.

Removing NIS+ That Was Installed Using NIS+ Commands

To remove NIS+ from a client machine that was set up as an NIS+ client using the `nisaddcred`, `domainname`, and `nisinit` commands as described in *NIS+ and DNS Setup and Configuration Guide*, perform the following steps:

1. Remove the `.rootkey` file.

```
clientmachine# rm -f /etc/.rootkey
```

2. Locate and kill the `keyserv`, `rpc.nisd`, `nis_cachemgr`, and `nscd` processes.

```
rootmaster# ps -ef | grep rpc.nisd
  root 137  1 67 16:34:44 ?  rpc.nisd
rootmaster# kill -9 137
rootmaster# ps -ef | grep keyserv
  root 714  1 67 16:34:44 ?  keyserv
rootmaster# kill -9 714
rootmaster# ps -ef | grep nis_cachemgr
  root 123  1 67 16:34:44 ?  nis_cachemgr
rootmaster# kill -9 123
rootmaster# ps -ef | grep nscd
  root 707  1 67 16:34:44 ?  nscd
rootmaster# kill -9 707
```

3. Remove the `/var/nis` directory and files.

```
rootmaster# rm -rf /var/nis/*
```

Removing NIS+ From a Server

This section describes how to remove NIS+ from an NIS+ server.

Keep in mind that removing NIS+ from a server does not remove the NIS+ name service from your network. See “Removing the NIS+ Namespace” on page 248 for information on removing the NIS+ name service from a network and returning to either NIS or `/etc` files for naming purposes.

Removing NIS+ From a Server

To remove NIS+ from a server, follow these steps:

1. Perform the steps necessary to remove NIS+ from a client.

An NIS+ server is also an NIS+ client. This means that you must first remove the client-related part of NIS+. You can use `nisclient -r` as described in “Removing NIS+ That Was Installed Using `nisclient`” on page 245 or the NIS+ command set as described in “Removing NIS+ That Was Installed Using NIS+ Commands” on page 246.

2. Remove the server’s `groups_dir` and `org_dir` directories.

```
server# nisrmdir -f groups_dir.domainname
server# nisrmdir -f org_dir.domainname
```

3. Locate and kill the `keyserv`, `rpc.nisd`, `nis_cachemgr`, and `nscd` processes on the server.

```
server# ps -ef | grep rpc.nisd
  root 137  1 67 16:34:44 ?  rpc.nisd
server# kill -9 137
server# ps -ef | grep keyserv
  root 714  1 67 16:34:44 ?  keyserv
server# kill -9 714
server# ps -ef | grep nis_cachemgr
  root 123  1 67 16:34:44 ?  nis_cachemgr
server# kill -9 123
server# ps -ef | grep nscd
  root 707  1 67 16:34:44 ?  nscd
server# kill -9 707
```

4. Remove the `/var/nis` directory and files.

```
rootmaster# rm -rf /var/nis/*
```

Removing the NIS+ Namespace

To remove the NIS+ namespace and return to using either NIS or `/etc` files for name services, follow these steps:

1. Remove the `.rootkey` file from the root master.

```
rootmaster# rm -f /etc/.rootkey
```

2. Remove the `groups_dir` and `org_dir` subdirectories from the root master root domain.

```
rootmaster# nisrmdir -f groups_dir.domainname  
rootmaster# nisrmdir -f org_dir.domainname
```

Where *domainname* is the name of the root domain, for example, `wiz.com`.

3. Remove the root domain.

```
rootmaster# nisrmdir -f domainname
```

Where *domainname* is the name of the root domain, for example, `wiz.com`.

4. Locate and kill the `keyserv`, `rpc.nisd`, `nis_cachemgr`, and `nscd` processes.

```
rootmaster# ps -ef | grep rpc.nisd
  root 137  1 67 16:34:44  ?  rpc.nisd
rootmaster# kill -9 137
rootmaster# ps -ef | grep keyserv
  root 714  1 67 16:34:44  ?  keyserv
rootmaster# kill -9 714
rootmaster# ps -ef | grep nis_cachemgr
  root 123  1 67 16:34:44  ?  nis_cachemgr
rootmaster# kill -9 123
rootmaster# ps -ef | grep nscd
  root 707  1 67 16:34:44  ?  nscd
rootmaster# kill -9 707
```

5. Create a new domain.

```
rootmaster# domainname name
```

Where *name* is the name of the new domain; for example, the name of the domain before you installed NIS+.

6. Remove the existing `/etc/defaultdomain` file.

```
rootmaster# rm /etc/defaultdomain
```

7. Recreate the `/etc/defaultdomain` file with the new domain name.

```
rootmaster# domainname > /etc/defaultdomain
```

8. Replace the original `nsswitch.conf` file.

If you set up this server with `nisserv -r`, you can use:

```
rootmaster# cp /etc/nsswitch.conf.no_nisplus /etc/nsswitch.conf
```

Alternatively, you can copy over one of the default switch template files. To use the default NIS switch file template, you would type:

```
rootmaster# cp /etc/nsswitch.nis etc/nsswitch.conf
```

To use the default `/etc/files` switch file template, you would type:

```
rootmaster# cp /etc/nsswitch.files etc/nsswitch.conf
```

9. Restart the `keyerv` process.

```
rootmaster# keyerv
```

10. Remove the `/var/nis` directory and files.

```
rootmaster# rm -rf /var/nis/*
```

11. Now restart your other name service (NIS or `/etc` files).

Part 3—Administering FNS

This part of the manual describes how to administer FNS.

<i>Administering FNS in NIS+</i>	<i>page 253</i>
<i>Federating NIS+ With Global Naming Systems</i>	<i>page 285</i>
<i>Administering the File System Namespace</i>	<i>page 291</i>
<i>Administering the Printer Namespace</i>	<i>page 301</i>

Administering FNS in NIS+

This chapter describes the setup and administration of the FNS implementation on top of the NIS+ environment. Use the following procedures for a standard setup (contexts are created for you). If you wish to set up contexts individually, then specific procedures in “Creating FNS Contexts Individually” on page 257 will apply.

<i>Estimating Resource Requirements</i>	<i>page 254</i>
<i>Setting Up NIS+ Service for FNS</i>	<i>page 254</i>
<i>Setting Up the FNS Namespace</i>	<i>page 255</i>
<i>Replicating FNS Service</i>	<i>page 256</i>

The following sections contain information to aid you in administering the FNS namespace after it has been set up.

<i>Managing and Examining FNS Contexts</i>	<i>page 267</i>
<i>Managing and Examining FNS Attributes</i>	<i>page 276</i>
<i>Maintaining Consistency Between NIS+ and FNS</i>	<i>page 278</i>
<i>Mapping FNS Contexts to NIS+ Objects</i>	<i>page 280</i>
<i>Browsing FNS Structures Using NIS+ Commands</i>	<i>page 280</i>
<i>Checking Access Control</i>	<i>page 282</i>
<i>Troubleshooting and Error Messages</i>	<i>page 284</i>

For an alphabetic listing of common FNS error messages, see Appendix B, “Error Messages.” For troubleshooting common FNS problems and solving them, see “FNS Problems and Solutions” on page 343.

Setting Up FNS

Setting up FNS involves preparing the NIS+ environment that FNS will use and then creating FNS contexts for organizations, users, hosts, services, and sites. Depending on the size of the organization, you should allow several hours for the FNS set up to be completed, not including the required hardware and software preparation or any NIS+ preparation.

Estimating Resource Requirements

Before proceeding with any installation procedure, you must first ensure that the machines on which NIS+ servers for supporting FNS will run have sufficient memory and disk storage.

For example, to support an FNS environment with 1200 users and hosts, you will need

- A minimum of 20 Mbytes of disk space beyond the space needed for NIS+
- An additional 40 Mbytes of swap space

Setting Up NIS+ Service for FNS

NIS+ objects used by FNS and standard NIS+ domain information should be supported on separate sets of servers. This avoids placing additional loads on the standard NIS+ service. It also allows you to keep the administration of FNS’s use of NIS+ and the standard NIS+ service separate.

All NIS+ objects used by FNS are kept under the `ctx_dir` directory of an NIS+ domain, at the same level as the domain’s `org_dir` directory. The NIS+ domain must be already set up before setting up FNS. That is, NIS+ domain tables such as `hosts` and `passwd` must already exist and be populated.

Before setting up the FNS namespace, do the following:

1. Set the `NIS_GROUP` environment variable to the name of the group that will be administering the FNS objects.

In fact, the `fncreate` command will not let you complete the FNS set up without setting the variable first. In this example, `NIS_GROUP` is set to `fns_admins.wiz.com`. When `fncreate` creates user and host contexts, they will be owned by those hosts and users, and not by the administrator who executed the command. Setting `NIS_GROUP` allows the administrators who are members of the group to subsequently modify these contexts even though they do not own the objects.

```
# set NIS_GROUP=fns_admins.wiz.com;export nis_group
```

2. To set up separate servers, create the `ctx_dir` directory for the NIS+ domain. Assign a master server to service it using the NIS+ command `nismkdir`.

The example shows how `nismkdir` creates the `ctx_dir` directory and assigns the machine `fns_mserver` to be the master server for that directory. Include the trailing dot as shown.

```
# nismkdir -m fns_mserver ctx_dir.wiz.com.
```

3. Use the `nisls` command to verify that the `ctx_dir` directory has been created.

```
# nisls wiz.com.  
ctx_dir  
groups_dir  
org_dir
```

Setting Up the FNS Namespace

The FNS namespace is created by the `fncreate` command. This command creates the contexts for the specified organization and all its subcontexts, including contexts for users and hosts in the NIS+ domain corresponding to the organization.

1. For a standard setup, use the syntax of `fncreate` as shown.

This creates the organization context for the root NIS+ domain, `wiz.com.`, contexts for all users found in the `passwd.org_dir` table, and contexts for all hosts found in the `hosts.org_dir` table in the `wiz.com` NIS+ domain.

```
# fncreate -t org org//
```

After setting up the FNS namespace, you should checkpoint the `ctx_dir` directory before performing other FNS operations.

2. Use `nisping` to checkpoint the `ctx_dir` directory:

```
# /usr/lib/nis/nisping -C ctx_dir.wiz.com.
```

For an organization with a few thousand users and hosts, the initial `fncreate` for an organization will typically take several hours; the subsequent checkpoint will also typically take several hours.

Replicating FNS Service

Additional replicas should be added to serve the `ctx_dir` directory after the FNS namespace has been set up on the master server. Replicas enhance availability and read performance of the servers.

1. Use the `nismkdir -s` command to add a replica `fns_rserver` for the `ctx_dir` directory and send the contents of the directory to the replica.

```
# nismkdir -s fns_rserver ctx_dir.wiz.com.
```

2. Checkpoint the `ctx_dir` directory periodically with the `nisping` command.

The recommended period is every few days. The period you choose depends on how frequently changes are made to the FNS namespace.

```
# /usr/lib/nis/nisping -C ctx_dir.wiz.com.
```

At this point, you are done with the initial FNS setup.

Creating FNS Contexts Individually

FNS contexts are created using the `fncreate(1M)` command. This section describes the `fncreate` command and its other options. Use this section to create FNS contexts individually rather than for the entire organization as described in the section “Setting Up the FNS Namespace” on page 255.

The `fncreate` command has the following syntax,

```
fncreate -t context_type [-f input_file][-o][-r reference_type][-s][-v] [-D] composite_name
```

where *context_type* specifies one of the following: `org`, `hostname`, `username`, `host`, `user`, `service`, `site`, `nsid`, `generic`, or `fs`.

`fncreate` creates a context of the specified type and binds it to the given composite name. It also creates subcontexts for the context.

When `fncreate` creates contexts, it also creates corresponding NIS+ directories and tables. These new directories and tables are managed by the same servers that manage the parent contexts of the new contexts, and can be administered using the same tools used to administer NIS+ entities.

Table 14-1 `fncreate` Command Options

Option	Description
-t	Specifies the type of context to create.
-f	Creates a context for every host or user listed in <i>input_file</i> . This option can only be used with the -t <code>username</code> or -t <code>hostname</code> option and is useful for creating contexts for a subset of users and hosts found in the corresponding NIS+ <code>passwd</code> and <code>hosts</code> tables, respectively.
-o	Creates only the context specified ¹ . Without the -o option, subcontexts are created according to the FNS policies.
-r	Specifies the <i>reference_type</i> of the generic context being created. It can only be used with the -t <code>generic</code> option.
-s	Creates new contexts for composite names already in use. Otherwise, no new contexts are created for names already bound.
-D	Displays information about the NIS+ object associated with a context each time a context is created. This option is useful for debugging.
-v	Displays information about the creation as each context is created.

1. The `org` context is the exception where the contexts for host name and user name are created but not populated.

When creating contexts bound to namespace identifiers, the name without the underscore (for example, `user`) is used to create the context and the name with the underscore (for example, `_user`) is then bound to the reference of the newly created context. This is done regardless of whether the name with or without the underscore is specified in the command line.

For example, the command

```
# fncreate -t username org/sales/_user
```

creates a context for `org/sales/user` and adds a binding for `org/sales/_user` to the context of `org/sales/user`.

Organization Context

Use the `org` type to create an organization context. The composite name must be that of an existing NIS+ domain. An NIS+ domain is an NIS+ directory with an `org_dir` subdirectory. Associated host and passwd tables for the domain must also exist.

Assume the root NIS+ domain is `wiz.com`. In the example

```
# fncreate -t org org/sales/
```

there must be an NIS+ domain named `sales`. When the new context is created, a `ctx_dir` directory, if it does not already exist, is created under the directory of the domain, `sales.wiz.com`. The previous example created an organization context for the composite name `org/sales/` and, in addition, created `hostname`, `username`, and `service` subcontexts for it, which in turn, created `host` and `user` contexts, and `service` subcontexts for hosts and users.

Effectively, the following commands are run:

```
fncreate -t hostname org/sales/host/  
fncreate -t username org/sales/user/  
fncreate -t service org/sales/service/
```

When `fncreate -o -t org` is used, only the organization context is created. The `hostname`, `username`, and `service` contexts are created but not populated with `host` and `user` contexts.

The `org` context is owned by the administrator who executed the `fncreate` command, as are the `hostname`, `username`, and `service` subcontexts. The `host` and `user` contexts, however, and their subcontexts are owned by the hosts or users for which the contexts were created. In order for the administrator to access `host` and `user` contexts, the `NIS_GROUP` environment variable must be set accordingly. See “Setting Up NIS+ Service for FNS” on page 254 for instructions.

All Hosts Context

The `hostname` type creates a `hostname` context in which host contexts can be created and bound. Host contexts and their subcontexts are created for each host name found in the NIS+ `hosts.org_dir` table unless the `-o` option is used. When the `-o` option is used, only the `hostname` context is created.

For example, running the command,

```
# fncreate -t hostname org/sales/host/
```

creates the `hostname` context and effectively runs the command:

```
# fncreate -t hostname org/sales/host/hname/
```

for each host name, *hname*, found in the `hosts.org_dir` table. It also adds a binding for `org/sales/_host/` that is bound to the reference of `org/sales/host/`.

The `hostname` context is owned by the administrator who executed the `fncreate` command. A host context and its subcontexts are owned by the host for which the contexts were created. That is, each host owns its own host context and subcontexts.

The `-f` option can be used to create contexts for a subset of the hosts found in the NIS+ table `hosts.org_dir`. It creates contexts for those hosts listed in the given input file.

Single Host Context

The `host` type creates the context and subcontexts for a host. The command automatically creates a `service` context for the host and a binding for `fs` unless the `-o` option is used. When the `-o` option is used, only the `host` context is created.

For example, the command

```
# fncreate -t host org/sales/host/capsule/
```

creates a context for the host named `capsule` and effectively runs the commands

```
fncreate -t service org/sales/host/capsule/service/  
fncreate -t fs org/sales/host/capsule/fs/
```

The host context and its subcontexts are owned by the host. In the above example, the host `capsule`, with NIS+ principle name `capsule.sales.wiz.com`, owns the contexts `org/sales/host/capsule/` and `org/sales/host/capsule/service/`.

The `hostname` context to which the host belongs must already exist. The host name supplied should already exist in the NIS+ `hosts.org_dir` table.

Host Aliases

Alias host names may exist in an NIS+ `hosts.org_dir` table. These appear in the table as a set of hosts with the same canonical name but different alias names.

In FNS, a single host with multiple alias names has a single host context. Alias names for that host in the host name context are bound to the reference of that host context.

All Users Context

The `username` type creates a `username` context in which user contexts can be created and bound. User contexts and their subcontexts are created for each user name found in the NIS+ `passwd.org_dir` table unless the `-o` option is used. When the `-o` option is used, only the `username` context is created.

For example, running the command

```
# fncreate -t username org/sales/user/
```

creates the `username` context and effectively runs the command

```
fncreate -t user org/sales/user/uname/
```

for each user, *uname*, that appears in the `passwd.org_dir` table. It also adds a binding for `org/sales/_user/` that is bound to the reference of `org/sales/user/`.

The username context is owned by the administrator who executed the `fncreate` command. A user context and its subcontexts are owned by the user for which the contexts were created. Each user owns his or her own `user` context and subcontexts.

The `-f` option can be used to create contexts for a subset of the users found in the NIS+ table `passwd.org_dir`. It creates contexts for those users listed in the given input file.

Single User Context

The `user` type creates the user context and subcontexts for a user. A `service` subcontext and a binding for `fs` are created under the `user` context unless the `-o` option is used. When the `-o` option is used, only the `user` context is created.

For example, the command

```
# fncreate -t user org/sales/user/jjones/
```

creates a `user` context for the user named `jjones` and effectively runs the commands

```
fncreate -t service org/sales/user/jjones/service/  
fncreate -t fs org/sales/user/jjones/fs/
```

The `user` context and its subcontexts are owned by the user for whom the contexts were created. In the above example, the contexts created are owned by the user `jjones` with NIS+ principal name `jjones.sales.wiz.com`.

The username context to which the user belongs must already exist. The user name supplied should already exist in the NIS+ `passwd.org_dir` table.

Service Context

The `service` type creates a service context in which service names can be bound. There is no restriction on what type of references may be bound in a service context. The policies depend on the applications that use the service context. For example, a group of desktop applications may bind references for a calendar, a rolodex, a fax service, and a printer in a service context.

For example, the command

```
# fncreate -t service org/sales/service/
```

creates a service context for the organization `sales`. Because the terminal atomic name is a namespace identifier, `fncreate` also adds a binding for `org/sales/_service/` that is bound to the reference of `org/sales/service/`. After executing this command, names such as `org/sales/service/calendar` and `org/sales/service/fax` can then be bound in this service context.

The service context supports a hierarchical namespace, with slash-separated left-to-right names, which allows an application to partition its namespace for different services. Continuing with the desktop applications example, a group of plotters may be named under the service context after the creation of the context.

```
# fncreate -t service org/sales/service/plotter
```

Names such as `org/sales/service/plotter/speedy` and `org/sales/service/plotter/production` could then be bound under the service context.

Note – Because the terminal atomic name is not a namespace identifier, no additional binding is added (as was the case with `service` and `_service`).

The service context created is owned by the administrator who ran the `fncreate` command.

Printer Context

The `printer` context is created under the `service` context of the respective composite name. Refer to Chapter 17, “Administering the Printer Namespace,” for more information.

Generic Context

The `generic` type creates a context for binding names used by applications.

A `generic` context is similar to a `service` context except it can have a different reference type. The `-r` option is used to specify the reference type for the `generic` context being created. If it is omitted, the reference type is inherited from its parent `generic` context, or if the parent context is not a `generic` context, the reference type used is a default `generic` reference type.

Like the `service` context, there is no restriction on what type of references may be bound in a `generic` context. The policies depend on the applications that use the `generic` context.

For example, the command

```
# fncreate -t generic -r WIDC_comm org/sales/service/extcomm
```

creates a `generic` context with the `WIDC_comm` reference type under the `service` context of the organization `sales`. Names such as `org/sales/service/extcomm/modem` can then be bound in this `generic` context.

The `generic` context supports a hierarchical namespace, with slash-separated left-to-right names, which allows an application to partition its namespace for different services. Continuing with the example above, a `generic` subcontext for `modem` can be created using the command

```
# fncreate -t generic org/sales/service/extcomm/modem
```

Names such as `org/sales/service/extcomm/modem/secure` and `org/sales/service/extcomm/modem/public` could then be bound under the `modem` context.

The generic context created is owned by the administrator who ran the `fncreate` command.

Site Context

The `site` type creates contexts in which site names can be bound.

For example, the command

```
# fncreate -t site org/sales/site/
```

creates a `site` context. Because the terminal atomic name is a namespace identifier, `fncreate` also adds a binding for `org/sales/_site/` that is bound to the reference of `org/sales/site/`.

The `site` context supports a hierarchical namespace, with dot-separated right-to-left names, which allows sites to be partitioned by their geographical coverage relationships.

For example, the commands

```
# fncreate -t site org/sales/site/east
# fncreate -t site org/sales/site/maynard.east
```

create a `site` context `east` and a `site` subcontext `maynard.east` for it.

Note – Because these terminal atomic names are not namespace identifiers, no additional binding are added (as was the case with `site` and `_site`).

The `site` context created is owned by the administrator who ran the `fncreate` command.

File Context

The `fs` type creates a file system context (or simply file context) for a user or a host. For example, the command

```
# fncreate -t fs org/sales/user/jjones/fs/
```

creates a file context for user `jjones`. Because the terminal atomic name is a namespace identifier, `fncreate` also adds a binding for `org/sales/user/jjones/_fs/` that is bound to the reference of `org/sales/user/jjones/fs/`.

The file context of a user is the user's home directory as it is stored in the NIS+ `passwd.org_dir` table. The file context of a host is the set of NFS file systems that the host exports.

Use the `fncreate_fs` command to create file contexts for organizations and sites, or to create file contexts other than the defaults available for users and hosts. See Chapter 16, "Administering the File System Namespace," for details.

The `fs` context created is owned by the administrator who ran the `fncreate` command.

Namespace Identifier Context

The `nsid` (namespace identifier) type creates a context in which namespace identifiers can be bound.

For example, the command

```
# fncreate -t nsid org/sales/site/maynard.east/
```

creates a `nsid` context for the site `maynard.east`, and permits the creation of subcontexts such as `service/`. Continuing with this example, you could then execute the command

```
# fncreate -t service org/sales/site/maynard.east/service/
```

to create a `service` context for `maynard.east`.

The `nsid` context created is owned by the administrator who ran the `fncreate` command.

Managing and Examining FNS Contexts

A number of tools are provided for examining and managing FNS contexts. The commands and their syntax are shown as follows. For additional information, see the man page for the tool. Note that `fnbind` has two usages.

```
fnlookup [-v][-L] composite_name
fnlist [-l][-v] [composite_name]
fnbind [-s][-v][-L] name new_name
fnbind -r [-s][-v] new_name [-O | -U] ref_type {[-O | -U] | address_type [-c|-x] address_contents}+
fnunbind composite_name
fnrename [-sv] context_name old_atomic_name new_atomic_name
fndestroy composite_name
```

Displaying the Binding

`fnlookup` displays the binding of the given composite name.

Table 14-2 `fnlookup` Command Options

Option	Description
-v	Displays the binding in more detail
-L	Displays the reference to which the XFN link is bound

For example, the command

```
# fnlookup user/jjones/
Reference type: onc_fn_user
Address type: onc_fn_nisplus
context type: user
```

shows the binding of the user `jjones`.

```
# fnlookup -v user/jjones/
Reference type: onc_fn_user
Address type: onc_fn_nisplus
length: 52
context type: user
representation: normal
version: 0
internal name: fns_user_jjones.ctx_dir.sales.wiz.com.
```

Suppose `user/James.Jones` is linked to `user/jjones`. The first command in the following example shows what `user/James.Jones` is bound to (an XFN link). The second command follows the XFN link, `user/jjones`, and shows what `user/jjones` is bound to (the user context).

```
# fnlookup user/James.Jones
Reference type: fn_link_ref
Address type: fn_link_addr
Link name: user/jjones

# fnlookup -L user/James.Jones
Reference type: onc_fn_user
Address type: onc_fn_nisplus
context type: user
```

Listing the Context

`fnlist` lists the contents of the context identified by the given name.

Table 14-3 `fnlist` Command Options

Option	Description
<code>-v</code>	Displays the binding in more detail
<code>-l</code>	Displays the bindings of the names bound in the named context

For example, the command

```
# fnlist user/  
Listing 'user/':  
jjones  
mladd  
jsmith  
James.Jones
```

shows the bindings under the `user` context.

If no name is given, the command lists the contents of the initial context.

```
# fnlist
Listing '':
_myorgunit
...
_myself
thishost
myself
_orgunit
_host
_thisens
myens
thisens
org
orgunit
thisuser
_thishost
myorgunit
_user
thisorgunit
host
_thisorgunit
_myens
user
```


When the `-l` option is given, the bindings of the names bound in the named context are displayed.

```
# fnlist -l user/
Listing bindings 'user/':
name: mladd
Reference type: onc_fn_user
Address type: onc_fn_nisplus
  context type: user
name: jsmith
Reference type: onc_fn_user
Address type: onc_fn_nisplus
  context type: user
name: James.Jones
Reference type: fn_link_ref
Address type: fn_link_addr
  Link name: user/jjones
name: jjones
Reference type: onc_fn_user
Address type: onc_fn_nisplus
  context type: user
```

When the `-v` option is given in conjunction with the `-l` option, the bindings are displayed in detail.

```
# fnlist -lv user/
Listing bindings 'user/':
name: mladd
Reference type: onc_fn_user
Address type: onc_fn_nisplus
  length: 52
  context type: user
  representation: normal
  version: 0
  internal name: fns_user_mladd.ctx_dir.sales.wiz.com.
name: jsmith
Reference type: onc_fn_user
Address type: onc_fn_nisplus
  length: 52
  context type: user
  representation: normal
  version: 0
  internal name: fns_user_jsmith.ctx_dir.sales.wiz.com.
name: James.Jones
Reference type: fn_link_ref
Address type: fn_link_addr
  length: 11
  data: 0x75 0x73 0x65 0x72 0x2f 0x6a 0x6a 0x6f 0x6e 0x65
user/jjones
name: jjones
Reference type: onc_fn_user
Address type: onc_fn_nisplus
  length: 52
  context type: user
  representation: normal
  version: 0
  internal name: fns_user_jjones.ctx_dir.sales.wiz.com.
```

Binding a Composite Name to a Reference

`fnbind` allows you to bind a composite name to a reference. There are two uses of this command. The first usage allows the user to bind the reference of an existing name to a new name. The second usage allows the user to bind a reference constructed using arguments in the command line to a name.

Table 14-4 `fnbind` Command Options

Option	Description
-s	Supersede any existing binding of the original composite name
-v	Prints out the reference used for the binding
-L	Creates an XFN link using <i>name</i> and binding it to <i>new_name</i>
-c	Stores address contents without XDR encoding
-x	Interprets address contents as a hexadecimal input string and store it as is
-r	Create a reference with a specified type and bind the reference to a name specified on the command line
-O	Interprets and stores type string as ASN.1 dot-separated integer list
-U	Interprets and stores type string as an DCE UUID

The first usage of `fnbind` is

```
fnbind [-s][-v][-L] name new_name
```

For example, the command

```
# fnbind myorgunit/service/printer user/mladd/service/printer
```

binds the name `user/mladd/service/printer` to the reference of `myorgunit/service/printer`.

If the given *new_name* is already bound, `fnbind -s` must be used or the operation will fail. In the above example, if `user/mladd/service/printer` is already bound, the `-s` option must be used to overwrite the existing binding with that of `myorgunit/service/printer`.

```
# fnbind -s myorgunit/service/printer user/mladd/service/printer
```

The `-v` option prints out the reference used for the binding.

```
# fnbind -v myorgunit/service/printer user/mladd/service/printer
Reference type: onc_printers
Address type: onc_fn_printer_nisplus
```

The following command constructs an XFN link out of `user/jjones` and binds it to the name `user/James.Jones`

```
# fnbind -L user/jjones user/James.Jones
```

Similarly, the following creates a link from `user/mladd/service/printer` to `myorgunit/service/printer`

```
# fnbind -sL myorgunit/service/printer user/mlad/service/printer
```

The second usage of `fnbind` constructs a reference using arguments in the command line and bounds it to the given name.

```
fnbind -r [-s] [-v] new_name [-O | -U] ref_type {[-O | -U] | address_type [-c|-x]address_contents}+
```

For example

```
# fnbind -r thisorgunit/service/calendar onc_calendar onc_cal_str staff@exodus
```

binds the name `thisorgunit/service/calendar` to the reference with type `onc_calendar` and address type `onc_cal_str` and address contents of `staff@exodus`.

By default, the address contents supplied in the command line is XDR-encoded before being stored in the reference. If the `-c` option is given, the address contents are stored in the clear, not as an XDR-encoded string. If the `-x` option is given, the address contents supplied in the command line are interpreted as a hexadecimal string and stored (and not XDR-encoded).

By default, the reference and address types of the reference to be constructed uses the `FN_ID_STRING` identifier format. If the `-O` option is given, the identifier format is `FN_ID_ISO_OID_STRING`, an ASN.1¹ dot-separated integer list string. If the `-U` option is given, the identifier format is `FN_ID_DCE_UUID`, a DCE² UUID in string form. For example, the following command binds to the name `thisorgunit/service/nx` a reference with OIDs as reference and address types and a hexadecimal string as the address contents.

```
# fnbind -r thisorgunit/service/nx -O 1.2.99.6.2.1 -O 1.2.99.6.2.3 -x ef12eab67290
```

Removing a Composite Name

`fnunbind` removes the given composite name from the namespace. Note that this does not remove the object associated with the name; it only unbinds the name from the object. For example, the command

```
# fnunbind user/jjones/service/printer/color
```

removes the binding associated with the name `user/jjones/service/printer/color`.

Renaming an Existing Binding

`fnrename` renames an existing binding. The following example renames the binding of `clndr` to `calendar`, in the context named by `user/jjones/service/`.

```
# fnrename user/jjones/service/ clndr calendar
```

The `-s` option is used to overwrite any binding to *new_atomic_name*.

1. See ISO 8824: 1990, Information Technology — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1)

2. See X/Open Preliminary Specification, October 1993, X/Open DCE: Remote Procedure Call (ISBN: 1-872630-95-2)

Destroying the Named Context

`fndestroy` removes the given composite name from the namespace and destroys the context named by the composite name.

For example, the command

```
# fndestroy user/jjones/
```

unbinds the name `user/jones/` from the namespace and destroys the context named by `user/jjones/`.

If the composite name identifies a context to be removed, the command fails if the context contains subcontexts.

Managing and Examining FNS Attributes

The `fnattr` command lets you update and examine attributes associated with FNS named objects. The four main options are for adding, deleting, listing, and modifying an attribute. In each of these cases, the identifier format is `FN_ID_STRING`, unless the option `-O` or `-U` is used.

Adding an Attribute

The `-a` option is for adding an attribute or adding a value to an attribute. You need to specify the composite name the attribute is associated with, the attribute identifier, and the values to add.

```
fnattr -a [-s] composite_name [-O | -U] identifier value1 [value2+]
```

The following example adds the attribute identifier `model` and the value `hplaser` to `thisorgunit/service/printer`.

```
# fnattr -as thisorgunit/service/printer model hplaser
```

Deleting an Attribute

To delete an attribute associated with an FNS named object, use the `-d` option. You can control what to delete:

- If an identifier is not specified, all the attributes associated with the named object are removed.
- If an identifier is specified, but without values, the entire attribute identified by *identifier* is removed.
- If individual values are specified, then only those values are removed from the attribute. Removal of the last value of an attribute is the same as removing the attribute itself.

```
fnattr -d composite_name [[-O | -U] identifier value1 [value2+]]
```

The following command deletes all the attributes associated with `thisorgunit/service/printer`.

```
# fnattr -d thisorgunit/service/printer
```

Listing an Attribute

The `-l` option is for listing attributes and their values. The command syntax is

```
fnattr -l composite_name [[-O | -U] identifier]
```

The following example lists the values of the `model` attribute of `thisorgunit/service/printer`.

```
# fnattr -l thisorgunit/service/printer model
laser
postscript
```

If an identifier is not specified, all the attributes associated with the named object are displayed.

Modifying an Attribute

The `-m` option lets you modify an attribute value. The command syntax is

```
fnattr -m composite_name [-O | -U] identifier old_value new_value
```

For example,

```
# fnattr -m thisorgunit/service/printer model postscript laser
```

replaces the value `postscript` with `laser`. Other attributes and values associated with `thisorgunit/service/printer` are not affected.

Other Options

The `-s` option means “add in supersede” mode. If an attribute with the specified identifier already exists, `-s` removes all of its values and replaces it with the value(s) added. If this option is omitted, the resulting values for the specified attribute includes the existing values and the new values added.

The `-O` option assumes the format of the attribute identifier is an ASN.1 dot-separated integer string list (`FN_ID_ISO_OID_STRING`).

The `-U` option assumes the format of the attribute identifier is a DCE UUID string form (`FN_ID_DCE_UUID`).

Maintaining Consistency Between NIS+ and FNS

A key task of the system administrator is to maintain consistency between FNS and NIS+. This means that the contents of FNS contexts and NIS+ tables must correspond. This correspondence is initially accomplished by the `fncreate` command, which ensures that FNS contexts are correctly created and populated and are consistent with NIS+ directory object and directory structures. After the FNS contexts have been set up, the correspondence needs to be maintained as new users and hosts are added to and removed from the system.

The Solstice AdminSuite product that adds user and host information to NIS+ also updates FNS. When updates to FNS or NIS+ are made independent of the Solstice AdminSuite, the resulting inconsistencies can be resolved by the use of the FNS tool, `fncheck`. `fncheck` checks for inconsistencies between user and host names in FNS, and user and host names in NIS+.

Checking Naming Inconsistencies

The `fncheck` command checks for naming inconsistencies between FNS `hostname` and `username` contexts, and the NIS+ standard system tables `hosts.org_dir` and `passwd.org_dir`, respectively. `fncheck` lists host and/or user names that are in the FNS namespace but are not in the NIS+ standard system tables. It also lists host or user names that are in the NIS+ standard system tables but not in the FNS namespace.

The command syntax is

```
fncheck[-r][-s][-u][-t hostname|username][domain_name]
```

Table 14-5 `fncheck` Command Options

Option	Description
-t	Specifies the type of context to check
-s	Lists host or user names from the NIS+ standard system tables that are not in the FNS namespace
-r	Lists host or user names from the FNS namespace that do not have entries in the corresponding NIS+ standard system tables
-u	Updates the FNS namespace based on information in the relevant NIS+ standard system tables

The `-t` option is used to check `hostname` and `username` contexts. For the `hostname` option, the `hostname` context associated with the organization is checked against the NIS+ `hosts.org_dir` table of the same organization. For the `username` option, the `username` context associated with the organization is checked against the NIS+ `passwd.org_dir` table in the same organization. When the `-t` option is omitted, both `hostname` and `username` contexts are checked.

The `-s` option does not list hosts or users that are already in the FNS namespace.

If the `-r` option is used in conjunction with the `-u` option, items that appear only in the FNS context are removed from the FNS context. If the `-s` option is used, items that appear only in the NIS+ standard system tables are added to the FNS context. If neither `-r` or `-s` are specified, items are added and removed from the FNS context to make it consistent with the corresponding NIS+ table.

Advanced FNS and NIS+ Issues

This section provides detailed information on the relationship between NIS+ objects and FNS objects. This information is useful when you must change the access control of FNS objects.

Mapping FNS Contexts to NIS+ Objects

FNS contexts are stored as NIS+ objects. All contexts associated with an organization are stored under the `ctx_dir` directory of the associated NIS+ domain, which resides at the same level as the `org_dir` directory of the same domain.

Use the `-v` option for the `fnlookup` or `fnlist` command to see the detailed description of references. The *internal name* field displays the name of the corresponding NIS+ object.

Browsing FNS Structures Using NIS+ Commands

The NIS+ command, `nislsl`, lists all the NIS+ objects used by FNS. For example, the following commands list the contents of the NIS+ domain directory and its `ctx_dir` subdirectory.

```
# nislsl wiz.com.  
wiz.com. :  
eng  
sales  
org_dir  
ctx_dir
```

```
# nislsl ctx_dir.wiz.com.  
ctx_dir_Wiz.COM.:  
fns  
fns_user  
fns_host  
fns_host_alto  
fns_host_mladd  
fns_host_elvira  
fns_user_jjones  
fns_user_jsmith  
fns_user_aw
```

List the contents of the `fns_hosts` table by using the `niscat` command.

```
# niscat fns_host.ctx_dir  
alto *BINARY* *BINARY*  
mladd *BINARY* *BINARY*  
elvira *BINARY* *BINARY*
```

Checking Access Control

Use `niscat -o` to see the access control of a context.

```
# niscat -o fns_host.ctx_dir
Object Name:fns_host
Owner: alto.wiz.com.
Group: admin.wiz.com.
Domain: ctx_dir.wiz.com.
Access Rights:r-c-rmcdrmcdr-c-
Time to Live:53:0:56
Object Type:TABLE
Table Type:H
Number of Columns:3
Character Separator:
Search Path:
Columns:
[0] Name:  atomicname
      Attributes:(SEARCHABLE, TEXTUAL DATA,CASE INSENSITIVE)
      Access Rights:-c-rmcdrmcdr-c-
[1] Name:  reference
      Attributes:(BINARY DATA)
      Access Rights:r-c-rmcdrmcdr-c-
[2] Name:  flags
      Attributes:(BINARY DATA)
      Access Rights:r-c-rmcdrmcdr-c-
```

To see the access control of a particular binding, use the name of the binding entry in the parent context's binding table (that is, the name displayed in the Internal Name field in the output of `fnlookup -v` and `fnlist -v`):

```
# niscat -o "[atomicname=alto],fns_host.ctx_dir"
Object Name:fns_host
Owner: alto.wiz.com.
Group: admin.wiz.com.
Domain: ctx_dir.wiz.com.
Access Rights:r-c-rmcdrmcdr-c-
Time to Live:12:0:0
Object Type:ENTRY
  Entry data of type H
  [1] - [5 bytes] 'alto'
  [2] - [104 bytes] '0x00 ...'
  [3] - [1 bytes] 0x01
```

To change the access control or ownership of a particular context, use the commands:

- `nischown`
- `nischmod`
- `nischgrp`

and give as an argument either the binding entry or the bindings table, depending on the object the operation is to effect.

Significance of Double Slashes

In the name, `org//`, the double slashes identifies the context of namespace identifiers associated with the root organizational name, as in `org//service/printer`.

In contrast, `org/` points to the root of the organizational context. Each `org` context has suborganizations as well as a pointer to context that contains namespace identifiers such as `service`, `user`, and `host`. `org/` names the root organizational context in which you can name suborganizations, as in `org/sales.finance`.

Significance of Trailing Slash

The trailing / names objects in the next naming system. You need it whenever you are going from one naming system to another. For example, the name, `org/sales.finance` names the context for naming suborganizations of the `sales.finance` organization, as in `org/audit.sales.finance`.

On the other hand, `org/sales.finance/` names the context for naming namespace identifiers of the `sales.finance` organization, as in `org/sales.finance/service/printer`.

Troubleshooting and Error Messages

For troubleshooting common FNS problems and solving them, see “FNS Problems and Solutions” on page 343

For example, a user attempted to create a context for `org//service/trading/bb`. The name `org//service/` was resolved successfully, but `trading` was not found in the context named by `org//service/`.

Thus, `trading/bb` is displayed as the part of the name that remains when the operation failed:

```
Error in creating 'org//service/trading/bb': Name Not Found:
'trading/bb'
```

In another example, a user attempted to destroy the context `org//service/dictionary/english`, but could not carry out the operation because the context named was not empty. The pair of single quotes (' ') indicates that FNS was able to resolve the complete name given, but could not complete the operation as requested:

```
Error in destroying 'org//service/dictionary/english': Context
Not Empty: ''
```

For an alphabetic listing of common FNS error messages, see Appendix B, “Error Messages.” For troubleshooting common FNS problems and solving them, see “FNS Problems and Solutions” on page 343.

Federating NIS+ With Global Naming Systems

15 

FNS supports federation of enterprise naming systems implemented using NIS+ into the global naming systems, DNS and X.500. This chapter describes the procedures for federating NIS+ with DNS and X.500. In general, the procedures involve

- Determining the NIS+ root reference for your NIS+ hierarchy
- Adding this information in the format required by the global naming system

Obtaining the NIS+ Root Reference

To federate NIS+ under DNS or X.500, information must be added to these respective naming systems to enable access to an NIS+ hierarchy from outside of the NIS+ hierarchy. This information comes from the NIS+ *root reference*, which consists of network address information describing how to reach the top of a particular NIS+ hierarchy.

The NIS+ root reference consists of a single address. The address has an address type of `onc_fn_nisplus_root` and contains a single, XDR-encoded string. The three items in the network address are separated by white spaces:

```
nis+_root_domain nis+_server [server_IP_address]
```

Table 15-1 is a description of the NIS+ root reference.

Table 15-1 NIS+ Root Reference

Address Element	Description
<code>nis+_root_domain</code>	The fully qualified name of the NIS+ root domain (trailing dot required)
<code>nis+_server</code>	The host name of one of the servers serving <i>nis+_root_domain</i>
<code>server_IP_address</code>	The IP address of <i>nis+_server</i> . This is optional if the address of <i>nis+_server</i> is already known. This means it is available through one of the naming services listed in the <code>/etc/nsswitch.conf</code> file.

In the following example,

```
wiz.com. wiz-nis-master
```

the address indicates that name of the NIS+ root domain is `wiz.com.` (trailing dot is significant), and that it can be reached using the host `wiz-nis-master`. The IP address of the server is not given because it is available through other means.

In another example,

```
woz.COM. wozwoz 133.33.33.33
```

indicates that the name of the NIS+ root domain is `woz.COM.` (trailing dot is significant) and that it can be reached using the host `wozwoz`, with the IP address `133.33.33.33`.

Federating NIS+ Under DNS

This section describes the steps required to add TXT (text) records for a subordinate enterprise naming system implemented with NIS+. To federate a subordinate naming system in DNS, you need to add reference information into DNS describing how to reach the subordinate naming system.

1. Obtain the NIS+ root reference for your NIS+ hierarchy, as described in “Obtaining the NIS+ Root Reference” on page 285.
2. Edit the DNS table (`/etc/named.local` is the default file name) and add a TXT record with the following format.

```
TXT "XFNNISPLUS nis+_root_domain nis+_server [server_IP_address]"
```

The following are examples of two records that convey the same information.

```
TXT "XFNNISPLUS wiz.com. nis-master"  
TXT XFNNISPLUS\ wiz.com.\ nis-master
```

The TXT record must be associated with a DNS domain that includes an NS (name server) record entry. The following is an example of a DNS table with reference information for NIS+ bound in it.

```
$ORIGIN Wiz.com  
@      IN SOA foo bar.eng.Wiz.com  
      (  
        100      ;; Serial  
        3600     ;; Refresh  
        3600     ;; Retry  
        3600     ;; Expire  
        3600     ;; Minimum  
      )  
      NS      nshost  
      TXT     "XFNNISPLUS wiz.com. wiz-nis-master 133.33.33.33"  
  
nshost IN  A 133.33.33.34
```

3. After adding the TXT record into the DNS table, either restart the DNS server, or send it a signal to reread the table.

```
# kill -HUP <pid of in.named>
```

Federating NIS+ Under X.500

In order to federate a subordinate naming system in X.500, you need to add reference information into X.500 describing how to reach the subordinate naming system. This section describes the steps for adding XFN reference information to the X.500 entry that will be the parent of the subordinate naming system.

1. Obtain the NIS+ root reference for your NIS+ hierarchy.

See “Obtaining the NIS+ Root Reference” on page 285.

2. Create an X.500 entry that supports XFN reference attributes.

For example, the following command creates a new X.500 entry called `c=us/o=wiz` with the object classes `top`, `organization`, and `XFN-supplement` (1.2.840.113536.25). The `XFN-supplement` object class allows the `c=us/o=wiz` entry to store reference information for a subordinate naming system.

```
# fnattr -a .../c=us/o=wiz object-class top organization XFN-supplement
```

If the X.500 entry already existed and was not defined with the `XFN-supplement` object class, it must be removed and re-created with the additional object class. Otherwise, it will not be able to hold reference information about the subordinate naming system.

3. Add the reference information about the subordinate NIS+ system to the entry.

After creating the X.500 entry, you can then add information about the subordinate NIS+ system by binding the appropriate NIS+ root reference to the named entry:

```
# fnbind -r ../c=us/o=wiz/ onc_fn_enterprise onc_fn_nisplus_root "wiz.com. bigbig"
```

This example binds the reference for the NIS+ hierarchy with the root domain name `wiz.com`, served by the machine `bigbig`, to the next naming system pointer (NNSP) of the X.500 entry `c=us/o=wiz`, thus linking the X.500 namespace with the `wiz.com` NIS+ namespace hierarchy.

The address format used is that of the NIS+ root reference described earlier. Note the use of the trailing slash in the name argument to `fnbind`, `../c=us/o=wiz/`, to signify that the reference is being bound to the NNSP of the entry, rather than to the entry itself."

Administering the File System Namespace

This chapter describes the file system namespace and the procedures for creating file contexts.

<i>The FNS File System Namespace</i>	<i>page 291</i>
<i>Creating File Contexts</i>	<i>page 294</i>
<i>Creating the Input File</i>	<i>page 295</i>
<i>Using Command-line Input</i>	<i>page 297</i>
<i>Administering File Contexts</i>	<i>page 299</i>

The FNS File System Namespace

Files may be named relative to users, hosts, organizations, and sites in FNS by appending the `fs` namespace identifier to the name of the object, and following this with the name of the file. For example, an engineering organization's tools directory might be named `org/engineering/fs/tools`.

The initial context is located under `/xfn` in the root directory. Thus a user might access the `tools` directory by typing

```
% cd /xfn/org/engineering/fs/tools
```

Existing applications can access this directory just as they would any other directory. Applications do not need to be modified in any way or use the XFN API.

NFS File Servers

NFS is Sun's distributed file system. The files associated with an object will generally reside on one or more remote NFS file servers. In the simplest case, the namespace identifier `fs` corresponds to the root of an exported NFS file system, as shown in Figure 16-1.

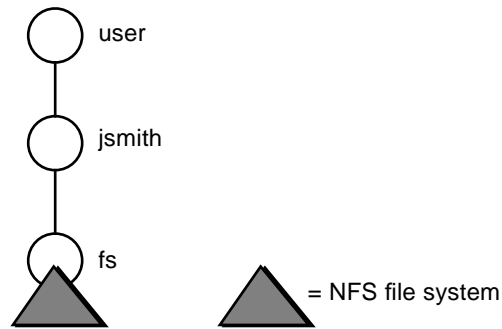


Figure 16-1 NFS File System—Simple Case

In contrast, an object's file system may be composed of multiple— and possibly overlapping—remote mounts, woven together into a “virtual” directory structure managed by FNS.

Figure 16-2 on page 293 illustrates how this capability might be used to piece together an organization's file system from three separate file servers. The `project` directory, along with its `lib` subdirectory, resides on one file server, while the `src` subdirectory resides on another. Users and applications need not be aware of the use of multiple servers; they see a single, seamless namespace.

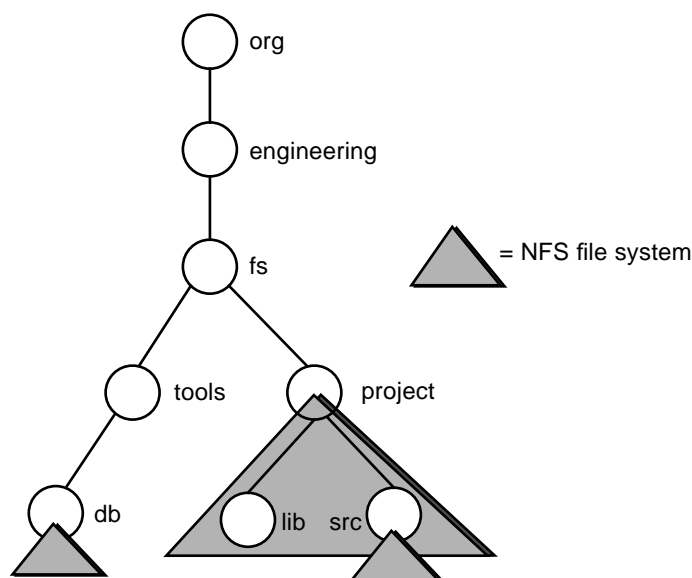


Figure 16-2 NFS File System—Multiple Servers

The Automounter

For efficiency, the automounter is used to mount FNS directories on demand. The default `/etc/auto_master` configuration file contains the line:

```
/xfn -xfn
```

which tells the automounter that the FNS namespace is “mounted” under `/xfn`, as specified by XFN.

Since the automounter is used to mount directories named through FNS, the subdirectories of an FNS directory cannot be listed until they have been mounted. For example, the command

```
% ls /xfn/org/engineering/fs
```

shows only those subdirectories that are currently mounted. To see the entire listing, use the `fnlist` command.

Creating File Contexts

The `fncreate_fs` command creates file contexts for organizations and sites. It may also be used to override the default file contexts for users and hosts that are created by the `fncreate` command.

There are two methods of using the `fncreate_fs` command. The context bindings may be provided by either the

- Input file – See “Creating the Input File” on page 295.
- Command line – See “Using Command-line Input” on page 297.

The two methods of `fncreate_fs` have the following syntax:

```
fncreate_fs -f input_file composite_name  
fncreate_fs [-v] [-r] composite_name [options][location...]
```

The `fncreate_fs` command-line options `-v` and `-r` are described in Table 16-1.

Table 16-1 `fncreate_fs` Command Options

Option	Description
-v	Sets verbose output, displaying information about the contexts being created and modified.
-r	Replaces the bindings in the context named by <i>composite_name</i> — and all of its subcontexts—with <i>only</i> those specified in the input. This is equivalent to destroying the context (and, recursively, its subcontexts), and then running <code>fncreate_fs</code> without this option. The <code>-r</code> option should be used with care.

The `fncreate_fs` command manipulates FNS contexts and bindings of the `onc_fn_fs` reference type. It uses an address of type `onc_fn_fs_mount` to represent each remote mount point. The data associated with an address of this type are the corresponding mount options and locations in a single string, XDR-encoded.

Creating the Input File

The input file supplies the names and values to be bound in the context of `composite_name`. Its format is based upon and similar, but not identical, to the format of indirect automount maps. The input file contains an entry with the form:

```
name [-options] [location . . .]
```

For each entry a reference to the location(s) and the corresponding options is bound to the name, `composite_name/name`.

The `name` field may be a simple atomic name or a slash-separated hierarchical name. It may also be “.” (dot), in which case the reference is bound directly to `composite_name`.

The `location` field specifies the host or hosts that serve the files for `composite_name/name`. In a simple NFS mount, `location` takes the form:

```
host: path
```

where `host` is the name of the server from which to mount the file system, and `path` is the path name of the directory to mount.

The `options` field is a comma-separated list of the mount options to use when mounting the directory. These options also apply to any subcontexts of `composite_name/name` that do not specify mount options of their own.

If `options` and `location` are both omitted, then no reference is bound to `composite_name/name`. Any existing reference is unbound.

Using the example from Figure 16-1 on page 292, suppose you want `jsmith`'s file system to be an NFS mount of the directory `/export/home/jsmith` from host `svr1`. The command would be run as follows:

```
% fncreate_fs -f infile user/jsmith/fs
```

with `infile` containing

```
.      svr1:/export/home/jsmith
```

To set up the file system illustrated in Figure 16-2 on page 293, run the command

```
% fncreate_fs -f infile org/engineering/fs
```

with `infile` containing

```
tools/db      svr1:/export/db
project       svr1:/export/proj
project/src   svr2:/export/src
```

To change the NFS mounts for `project` and its subcontext to be read-only, you can change `infile` as follows:

```
tools/db      svr1:/export/db
project -ro   svr1:/export/proj
project/src   svr2:/export/src
```

The `-ro` is unnecessary in the third line. Since `src` is a subcontext of `project`, it will inherit the `-ro` mount option from above.

The following input file would make all of the mounts read-only except for `org/engineering/fs/project/src`.

```
.           -ro
tools/db    svr1:/export/db
project     svr1:/export/proj
project/src -rw  svr2:/export/src
```

Using Command-line Input

The `fncreate_fs` command also allows the binding description to be provided on the command line:

```
fncreate_fs composite_name [options] [location ...]
```

This is equivalent to using the first form of the command and providing a one-line input file containing “.” in the *name* field, and the given mount options and locations. The previous example in which `jsmith`’s file system was set could be set from the command line as follows:

```
% fncreate_fs user/jsmith/fs svr1:/export/home/jsmith
```

Similarly, the hierarchy in Figure 16-2 on page 293 could have been set up by running the sequence of commands:

```
% fncreate_fs org/engineering/fs/tools/db svr1:/export/db
% fncreate_fs org/engineering/fs/project svr1:/export/proj
% fncreate_fs org/engineering/fs/project/src svr2:/export/src
```

To make all three of the mounts read-only, you would run this command:

```
% fncreate_fs org/engineering/fs -ro
```

Advanced Input Formats

The following two sections apply to both input file and command-line input formats.

Multiple Locations

Multiple *location* fields may be specified for NFS file systems that are exported from multiple, functionally equivalent locations:

```
% fcreate_fs org/sales/fs svr1:/sales svr2:/sales
```

The automounter will attempt to choose the best server from among the alternatives provided. If several locations in the list share the same path name, they may be combined using a comma-separated list of host names:

```
% fcreate_fs org/sales/fs svr1,svr2:/sales
```

The hosts may be weighted, with the weighting factor appended to the host name as a nonnegative integer in parentheses: the lower the number, the more desirable the server. The default weighting factor is zero (most desirable).

The following example illustrates one way to indicate that *svr2* is the preferred server:

```
% fcreate_fs org/sales/fs svr1(2),svr2(1):/sales
```

Variable Substitution

Variable names, prefixed by \$, may be used in the *options* or *location* fields of *fcreate_fs*. For example, a location may be given as:

```
svr1:/export/$CPU
```

The automounter will substitute client-specific values for these variables when mounting the corresponding file systems. In the above example, `$CPU` is replaced by the output of `uname -p`; for example, `sparc`.

Backward Compatibility Input Format

For additional compatibility with automount maps, the following input file format is also accepted by `fncreate_fs`:

```
name                [options] [location ...] \  
    /offset1         [options1] location1 ... \  
    /offset2         [options2] location2 ... \  
    ...
```

where each *offset* field is a slash-separated hierarchy. The backslash (“\”) indicates the continuation of a single long line. This is interpreted as being equivalent to:

```
name                [options] [location ...]  
name/offset1       [options1] location1 ...  
name/offset2       [options2] location2 ...  
...
```

The first line is omitted if both *options* and *location* are omitted. This format is for compatibility only. It provides no additional functionality, and its use is discouraged.

Administering File Contexts

File contexts may be inspected using the `fnlist` and `fnlookup` commands, and may be pruned or destroyed using `fnunbind` and `fndestroy`. These commands and sample output are described in Chapter 14, “Administering FNS in NIS+.” Refer also to the man page for each command.

Administering the Printer Namespace

This chapter describes the administration of the printer namespace. The `printer` context is not part of the XFN policies. It is provided in FNS in order to store printer bindings.

The Printer Namespace

FNS provides the capability to store printer bindings in the FNS namespace. This gives print servers the means to advertise their services and allow users to browse and choose amongst the available printers.

Printer bindings are stored in `printer` contexts, which are associated with organizations, users, hosts, and sites. Hence, each organization, user, host, and site has its own `printer` context.

The `printer` context is created under the `service` context of the respective composite name. For example, the composite name shown below has the following `printer` context:

```
org/wiz.com/service/printer
```

The name of a printer for a host, `labpc`, with a `printer` context might look like this:

```
host/labpc/server/printer/laser
```

Administering printer Contexts

Currently, `printer` contexts are supported for name service of files, NIS, and NIS+. The manner in which the bindings are stored in the `printer` context varies according to the underlying name service used for implementing FNS. Printer bindings are associated with organizations. For NIS and files, only printer bindings are stored. NIS+, however, stores the printer bindings in the `printer` context, which allows the printer namespace to be arranged hierarchically and be associated with `org`, `host`, `user`, and `site` contexts.

Using Files

Files are used as the default name service if neither NIS nor NIS+ is present. The printer bindings are stored in the `/etc/printers.conf` file, which is the printer configuration database used to describe printers. Each printer binding requires its own entry in this file. For example, if you have a printer named `printer1`, with the alias `ps`, you would add an entry to the `printers.conf` file in this format:

```
printer1|ps:bsdaddr=server_name
```

In this example, when a look up is performed on an address containing `printer1`, the address type of `onc_printers_bsdaddr` is returned. For more information about the required file format in `printers.conf`, see the `printers.conf(4)` man page.

Using NIS

If NIS is the underlying name service, the map that is used to store the printer configuration is called `printers.conf.byname`. Each printer binding has an entry in this file. For example, if you have a printer named `printer2`, with the alias `lp`, you would add an entry to the `printers.conf.byname` file in this format:

```
printer2|lp:bsdaddr=server_name
```

For more information about the syntax required, see the `printers.conf.byname(4)` man page.

Using NIS+

If NIS+ is the underlying name service for FNS, administering the printer contexts is simplified by the `fncreate_printer` command, which creates the printer context for organization, users, hosts and sites.

The `fncreate_printer` command takes the following arguments:

```
fncreate_printer composite_name printer_name printer_address
```

where `printer_address` is in the form `addresstype=address`. In the next example, the `printer_address` is `bsdaddr=labpc`. For more information, see the `fncreate_printer(1)` man page.

In this example, a printer binding for the printer `laser-jet` for the user `jsmith` is created:

```
% fncreate_printer user/jsmith laser-jet bsdaddr=labpc
```

The new binding, `user/jsmith/service/printer/laser-jet`, has the address type `onc_printers_bsdaddr`, and the address `labpc`. FNS adds the prefix `onc_printers_` to the address type.

In NIS+, it is possible to organize printers hierarchically. For example, printers can be listed under the printer context, as shown by the following commands:

```
% fncreate_printer org/wiz.com color/lpq bsdaddr=colorful
% fncreate_printer org/wiz.com color/laser bsdaddr=colorprt
% fncreate_printer org/wiz.com color/inkjet bsdaddr=colorjet
```

The `fncreate` command added the printer bindings for the printers, `lpq`, `laser`, and `inkjet` to the context `color` present under the printer context. The result looks like this:

```
org/wiz.com/service/printer/color/lpq
org/wiz.com/service/printer/color/laser
org/wiz.com/service/printer/color/inkjet
```

Similarly, color printers, green, red and blue for user jsmith can be organized as follows:

```
user/jsmith/service/printer/color/green
user/jsmith/service/printer/color/red
user/jsmith/service/printer/color/blue
```

Printer bindings (contexts) in NIS+ can be removed using the `fndestroy` command. For example, to remove the printer context in this example, use the command:

```
% fndestroy user/jsmith/service/printer/laser-jet
```

Part 4—Appendices

This part of the manual provides reference material.

<i>Problems and Solutions</i>	<i>page 307</i>
<i>Error Messages</i>	<i>page 349</i>
<i>Information in NIS+ Tables</i>	<i>page 403</i>

Problems and Solutions



This appendix describes some of the problems you may encounter while administering an NIS+ namespace. Problems are grouped according to type. For each problem there is a list of common symptoms, a description of the problem, and one or more suggested solutions.

In addition to this appendix, there is an appendix containing an alphabetic listing of the more common NIS+ error messages. If you are responding to a specific error message, check Appendix B, “Error Messages” first. If the problem is simple, or specific to a single error message, its solution is usually described in Appendix B.

This chapter covers the following types of NIS+ problems:

<i>Namespace Administration Problems</i>	<i>page 308</i>
<i>Namespace Database Problems</i>	<i>page 311</i>
<i>NIS Compatibility Problems</i>	<i>page 312</i>
<i>Object Not Found Problems</i>	<i>page 314</i>
<i>Ownership and Permission Problems</i>	<i>page 318</i>
<i>Security Problems</i>	<i>page 321</i>
<i>Slow Performance and System Hang Problems</i>	<i>page 332</i>
<i>System Resource Problems</i>	<i>page 337</i>
<i>User Problems</i>	<i>page 338</i>
<i>Other NIS+ Problems</i>	<i>page 341</i>
<i>FNS Problems and Solutions</i>	<i>page 343</i>

Namespace Administration Problems

This section describes problems that may be encountered in the course of routine namespace administration work.

Symptoms

- Illegal object type for operation message.
- Other “object problem” error messages
- Initialization failure
- Checkpoint failures
- Difficulty adding a user to a group
- Logs too large/lack of disk space/difficulty truncating logs
- Cannot delete `groups_dir` or `org_dir`

Illegal Object Problems

Symptoms

- Illegal object type for operation message
- Other “object problem” error messages

Possible Causes

There are a number of possible causes for this error message:

- You have attempted to create a table without any searchable columns.
- A database operation has returned the status of `DB_BADOBJECT` (see the `nis_db` man page for information on the db error codes).
- You are trying to add or modify a database object with a length of zero.
- You attempted to add an object without an owner.
- The operation expected a directory object, and the object you named was not a directory object.
- You attempted to link a directory to a LINK object.
- An object that was not a group object was passed to the `nisgrpadm` command.
- An operation on a group object was expected, but the type of object specified was not a group object.

- An operation on a table object was expected, but the object specified was not a table object.

`nisinit` *Fails*

Make sure that:

- You can ping the NIS+ server to check that it is up and running as a machine.
- The NIS+ server that you specified with the `-H` option is a valid server and that it is running the NIS+ software.
- `rpc.nisd` is running on the server.
- The nobody class has read permission for this domain.
- The netmask is properly set up on this machine.

Checkpoint Keeps Failing

If checkpoint operations with a `nisping -C` command consistently fail, make sure you have sufficient swap and disk space. Check for error messages in `syslog`. Check for core files filling up space.

Cannot Add User to a Group

A user must first be an NIS+ principal client with a LOCAL credential in the domain's `cred` table before the user can be added as a member of a group in that domain. A DES credential alone is not sufficient.

Logs Grow too Large

Failure to regularly checkpoint your system with `nisping -C` causes your log files to grow too large. Logs are not cleared on a master until *all* replicas for that master are updated. If a replica is down or otherwise out of service or unreachable, the master's logs for that replica cannot be cleared. Thus, if a replica is going to be down or out of service for a period of time, you should remove it as a replica from the master with `nisrmdir -f -s` for all directories, including `groups_dir` and `org_dir`.

Lack of Disk Space

Lack of sufficient disk space will cause a variety of different error messages. (See “Insufficient Disk Space” on page 338 for additional information.)

Cannot Truncate Transaction Log File

First, check to make sure that the file in question exists and is readable and that you have permission to write to it.

- You can use `ls -l /var/nis/trans.log` to display the transaction log.
- You can use `nislsl -l` and `niscat` to check for existence, permissions, and readability.
- You can use `syslog` to check for relevant messages.

The most likely cause of inability to truncate an existing log file for which you have the proper permissions is lack of disk space. (The checkpoint process first creates a duplicate temporary file of the log before truncating the log and then removing the temporary file. If there is not enough disk space for the temporary file, the checkpoint process cannot proceed.) Check your available disk space and free up additional space if necessary.

Domain Name Confusion

Domain names play a key role in many NIS+ commands and operations. To avoid confusion, you must remember that except for root servers, all NIS+ masters and replicas are clients of the domain *above* the domain that they serve. If you make the mistake of treating a server or replica as if it were a client of the domain that it serves, you may get `Generic system error or Possible loop detected in namespace directoryname: domainname` error messages.

For example, the machine `aladin` might be a client of the `subwiz.wiz.com.` domain. If the master server of the `subwiz.wiz.com.` subdomain is the machine `merlin`, then `merlin` is a client of the `wiz.com.` domain. When using, specifying, or changing domains, remember these rules to avoid confusion:

1. Client machines belong to a given domain or subdomain.

2. Servers and replicas that serve a given subdomain are clients of the domain above the domain they are serving.
3. The only exception to Rule 2 is that the root master server and root replica servers are clients of the same domain that they serve. In other words, the root master and root replicas are all clients of the root domain.

Thus, in the example above, the fully qualified name of the `aladin` machine is `alladin.subwiz.wiz.com`. The fully qualified name of the `merlin` machine is `merlin.wiz.com`. The name `merlin.subwiz.wiz.com` is wrong and will cause an error because `merlin` is a client of `wiz.com`, not `subwiz.wiz.com`.

Cannot Delete `org_dir` or `groups_dir`

Always delete `org_dir` and `groups_dir` *before* deleting their parent directory. If you use `nisrmdir` to delete the domain before deleting the domain's `groups_dir` and `org_dir`, you will not be able to delete either of those two subdirectories.

Namespace Database Problems

This section covers problems related to the namespace database and tables.

Symptoms

Error messages with operative clauses such as:

- “Abort_transaction: Internal database error”
- “Abort_transaction: Internal Error, log entry corrupt”
- “Callback: - select failed”
- “CALLBACK_SVC: bad argument”

See also “Ownership and Permission Problems” on page 318.

Multiple `rpc.nisd` Parent Processes

Symptoms

Various Database and transaction log corruption error messages containing the terms:

- “Corrupt log”

- “Log corrupted”
- “Log entry corrupt”
- “Corrupt database”
- “Database corrupted”

Possible Causes

You have multiple *independent* `rpc.nisd` daemons running. In normal operation, `rpc.nisd` may spawn other child `rpc.nisd` daemons. This causes no problem. However, if two parent `rpc.nisd` daemons are running at the same time on the same machine, they will overwrite each other’s data and corrupt logs and databases. (Normally, this could only occur if someone started running `rpc.nisd` by hand.)

Diagnosis

Run `ps -ef | grep rpc.nisd`. Make sure that you have no more than one parent `rpc.nisd` process.

Solution

If you started `rpc.nisd` with the `-B` option, you must also kill the `rpc.nisd_resolv` daemon.

If you have more than one “parent” `rpc.nisd` entries, you must kill all but one of them. Use `kill -9 process-id`, then run the `ps` command again to make sure it has died.

If an NIS+ database is corrupt, you will also have to restore it from your most recent backup that contains an uncorrupted version of the database. You can then use the logs to update changes made to your namespace since the backup was recorded. However, if your logs are also corrupted, you will have to recreate by hand any namespace modifications made since the backup was taken.

NIS Compatibility Problems

This section describes problems having to do with NIS compatibility with NIS+ and earlier systems and the switch configuration file.

Symptoms

The `nsswitch.conf` file fails to perform correctly.

Error messages with operative clauses such as:

- “Unknown user”
- “Permission denied”
- “Invalid principal name”

User Cannot Log In After Password Change

Symptoms

New users, or users who recently changed their password are unable to log in at all, or able to log in on one or more machines but not on others. The user may see error messages with operative clauses such as:

- “Unknown user: *username*”
- “Permission denied”
- “Invalid principal name”

First Possible Cause

Password was changed on NIS machine.

If a user or system administrator uses the `passwd` command to change a password on a Solaris 2.x machine running NIS in a domain served by NIS+ namespace servers, the user’s password is changed only in that machine’s `/etc/passwd` file. If the user then goes to some other machine on the network, the user’s new password will not be recognized by that machine. The user will have to use the old password stored in the NIS+ `passwd` table.

Diagnosis

Check to see if the user’s old password is still valid on another NIS+ machine.

Solution

Use `passwd` on a machine running NIS+ to change the user’s password.

Second Possible Cause

Password changes take time to propagate through the domain.

Diagnosis

Namespace changes take a measurable amount of time to propagate through a domain and an entire system. This time might be as short as a few seconds or as long as many minutes, depending on the size of your domain and the number of replica servers.

Solution

You can simply wait the normal amount of time for a change to propagate through your domain(s). Or you can use the `nisping org_dir` command to resynchronize your system.

`nsswitch.conf` ***File Fails to Perform Correctly***

A modified (or newly installed) `nsswitch.conf` file fails to work properly.

Symptoms

You install a new `nsswitch.conf` file or make changes to the existing file, but your system does not implement the changes.

Possible Cause

Each time an `nsswitch.conf` file is installed or changed, you must reboot the machine for your changes to take effect. This is because `nscd` caches the `nsswitch.conf` file.

Solution

Check your `nsswitch.conf` file against the information contained in the `nsswitch.conf` man page. Correct the file if necessary, and then reboot the machine.

Object Not Found Problems

This section describes problem in which NIS+ was unable to find some object or principal.

Symptoms

Error messages with operative clauses such as:

- “Not found”
- “Not exist”
- “Can’t find suitable transport for *name*”
- “Cannot find”
- “Unable to find”
- “Unable to stat”

Syntax or Spelling Error

The most likely cause of some NIS+ object not being found is that you mistyped or misspelled its name. Check the syntax and make sure that you are using the correct name.

Incorrect Path

A likely cause of an “*object not found*” problem is specifying an incorrect path. Make sure that the path you specified is correct. Also make sure that the NIS_PATH environment variable is set correctly.

Domain Levels Not Correctly Specified

Remember that all servers are clients of the domain above them, not the domain they serve. There are two exceptions to this rule:

- The root masters and root replicas are clients of the root domain.
- NIS+ domain names end with a period. When using a fully qualified name you must end the domain name with a period. If you do not end the domain name with a period, NIS+ assumes it is a partially qualified name. However, the domain name of a machine should not end with a dot in the `/etc/defaultdomain` file. If you add a dot to a machine’s domain name in the `/etc/defaultdomain` file, you will get `Could not bind to server serving domain name` error messages and encounter difficulty in connecting to the net on boot up.

Object Does Not Exist

The NIS+ object may not have been found because it does not exist, either because it has been erased or not yet created. Use `nislsl -l` in the appropriate domain to check that the object exists.

Lagging or Out-of-Sync Replica

Unlike NIS, there is no binding with NIS+.

When you create or modify an NIS+ object, there is a time lag between the completion of your action and the arrival of the new updated information at a given replica. In ordinary operation, namespace information may be queried from a master or any of its replicas. A client automatically distributes queries among the various servers (master and replicas) to balance system load. This means that at any given moment you do not know which machine is supplying you with namespace information. If a command relating to a newly created or modified object is sent to a replica that has not yet received the updated information from the master, you will get an “object not found” type of error or the old out-of-date information. Similarly, a general command such as `nislsl` may not list a newly created object if the system sends the `nislsl` query to a replica that has not yet been updated.

You can use `nisping` to resync a lagging or out of sync replica server.

Alternatively, you can use the `-M` option with most NIS+ commands to specify that the command must obtain namespace information from the domain’s master server. In this way you can be sure that you are obtaining and using the most up-to-date information. (However, you should use the `-M` option only when necessary because a main point of having and using replicas to serve the namespace is to distribute the load and thus increase network efficiency.)

Files Missing or Corrupt

One or more of the files in `/var/nis/data` directory has become corrupted or erased. Restore these files from your most recent backup.

Old /var/nis Filenames

In Solaris 2.4 and earlier, the `/var/nis` directory contained two files named `hostname.dict` and `hostname.log`. It also contained a subdirectory named `/var/nis/hostname`. In Solaris 2.5, the two files are renamed `trans.log` and `data.dict`, and the subdirectory is named `/var/nis/data`.

Do not rename the `/var/nis` or `/var/nis/data` directories or any of the files in these directories that were created by `nisinit` or any of the other NIS+ setup procedures.

In Solaris 2.5, the *content* of the files has also been changed and they are not backward compatible with Solaris 2.4 or earlier. Thus, if you rename either the directories or the files to match the Solaris 2.4 patterns, the files will not work with either the Solaris 2.4 or the Solaris 2.5 version of `rpc.nisd`. Therefore, you should not rename either the directories or the files.

Blanks in Name

Symptoms

Sometimes an object is there, sometimes it is not. Some NIS+ or UNIX commands report that an NIS+ object does not exist or cannot be found, while other NIS+ or UNIX commands do find that same object.

Diagnoses:

Use `nislsl` to display the object's name. Look carefully at the object's name to see if the name actually begins with a blank space. (If you accidentally enter two spaces after the flag when creating NIS+ objects from the command line with NIS+ commands, some NIS+ commands will interpret the second space as the beginning of the object's name.)

Solution

If an NIS+ object name begins with a blank space, you must either rename it without the space or remove it and then recreate it from scratch.

Cannot Use Automounter

Symptoms

You cannot change to a directory on another host.

Possible Cause

Under NIS+, automounter names must be renamed to meet NIS+ requirements. NIS+ cannot access `/etc/auto*` tables that contain a period in the name. For example, NIS+ cannot access a file named `auto.direct`.

Diagnosis

Use `nislsl` and `niscat` to determine if the automounter tables are properly constructed.

Solution

Change the periods to underscores. For example, change `auto.direct` to `auto_direct`. (Be sure to change other maps that might reference these.)

Ownership and Permission Problems

This section describes problems related to user ownership and permissions.

Symptoms

Error messages with operative clauses such as:

- “Unable to stat name”
- “Unable to stat NIS+ directory name”
- “Security exception on LOCAL system”
- “Unable to make request”
- “Insufficient permission to . . .”
- “You name do not have secure RPC credentials”

Another Symptom:

- User or root unable to perform any namespace task.

No Permission

The most common permission problem is the simplest: you have not been granted permission to perform some task that you try to do. Use `niscat -o` on the object in question to determine what permissions you have. If you need additional permission, you, the owner of the object, or the system administrator can either change the permission requirements of the object (as described in Chapter 7, “Administering NIS+ Access Rights,”) or add you to a group that does have the required permissions (as described in Chapter 9, “Administering NIS+ Groups”).

No Credentials

Without proper credentials for you and your machine, many operations will fail. Use `nismatch` on your home domain’s `cred` table to make sure you have the right credentials. See “Corrupted Credentials” on page 327 for more on credentials-related problems.

Server Running at Security Level 0

A server running at security level 0 does not create or maintain credentials for NIS+ principals.

If you try to use `nispaswd` on a server that is running at security level 0, you will get the error message: You *name* do not have secure RPC credentials in NIS+ domain *name*.

Security level 0 is only to be used by administrators for initial namespace setup and testing purposes. Level 0 should not be used in any environment where ordinary users are active.

User Login Same as Machine Name

A user cannot have the same login ID as a machine name. When a machine is given the same name as a user (or vice versa), the first principal can no longer perform operations requiring secure permissions because the second principal’s key has overwritten the first principal’s key in the `cred` table. In addition, the second principal now has whatever permissions were granted to the first principal.

For example, suppose a user with the login name of `pine` is granted namespace read-only permissions. Then a machine named `pine` is added to the domain. The user `pine` will no longer be able to perform any namespace operations requiring any sort of permission, and the root user of the machine `pine` will only have read-only permission in the namespace.

Symptoms

- The user or machine gets “permission denied” error messages.
- Either the user or root for that machine cannot successfully run `keylogin`.
- Security exception on LOCAL system. `UNABLE TO MAKE REQUEST. error message.`
- If the first principal did not have read access, the second principal might not be able to view otherwise visible objects.

Note – When running `nisclient` or `nisaddcred`, if the message `Changing Key` is displayed rather than `Adding Key`, there is a duplicate user or host name already in existence in that domain.

Diagnosis

Run `nismatch` to find the host and user in the `hosts` and `passwd` tables to see if there are identical host names and user names in the respective tables:

```
nismatch username passwd.org_dir
```

Then run `nismatch` on the domain’s `cred` table to see what type of credentials are provided for the duplicate host or user name. If there are both LOCAL and DES credentials, the `cred` table entry is for the user; if there is only a DES credential, the entry is for the machine.

Solution

Change the machine name. (It is better to change the machine name than to change the user name.) Then delete the machine’s entry from the `cred` table and use `nisclient` to reinitialize the machine as an NIS+ client. (If you wish, you can use `nistbladm` to create an alias for the machine’s old name in the `hosts` tables.) If necessary, replace the user’s credentials in the `cred` table.

Bad Credentials

See “Corrupted Credentials” on page 327.

Security Problems

This section describes common password, credential, encryption, and other security-related problems.

Symptoms

Error messages with operative clauses such as:

- “Authentication error”
- “Authentication denied”
- “Cannot get public key”
- “Chkey failed”
- “Insufficient permission to”
- “Login incorrect”
- “Keyserv fails to encrypt”.
- “No public key”
- “Permission denied”
- “Password [problems]”

User or root unable to perform any namespace operations or tasks. (See also “Ownership and Permission Problems” on page 318.)

“Login Incorrect” Message

The most common cause of a “login incorrect” message is the user mistyping the password. Have the user try it again. Make sure the user knows the correct password and understands that passwords are case-sensitive and that the letter “o” is not interchangeable with the numeral “0,” nor is the letter “l” the same as the numeral “1.”

Other possible causes of the “login incorrect” message are:

- The password has been locked by an administrator. See “Locking a Password” on page 157 and “Unlocking a Password” on page 158.

- The password has been locked because the user has exceeded an inactivity maximum. See “Specifying Maximum Number of Inactive Days” on page 166.
- The password has expired. See “Password Privilege Expiration” on page 164.

See Chapter 8, “Administering Passwords” for general information on passwords.

Password Locked, Expired, or Terminated

A common cause of a “Permission denied, password expired,” type message is that the user’s password has passed its age limit or the user’s password privileges have expired. See Chapter 8, “Administering Passwords” for general information on passwords.

- See “Setting a Password Age Limit” on page 160.
- See “Password Privilege Expiration” on page 164.

Stale and Outdated Credential Information

Occasionally, you may find that even though you have created the proper credentials and assigned the proper access rights, some client requests still get denied. This may be due to out-of-date information residing somewhere in the namespace.

Storing and Updating Credential Information

Credential-related information, such as public keys, is stored in many locations throughout the namespace. NIS+ updates this information periodically, depending on the time-to-live values of the objects that store it, but sometimes,

between updates, it gets out of sync. As a result, you may find that operations that should work, don't work. Table A-1 lists all the objects, tables, and files that store credential-related information and how to reset it.

Table A-1 Where Credential-Related Information is Stored

Item	Stores	To Reset or Change
cred table	NIS+ principal's secret key and public key. These are the master copies of these keys.	Use <code>nisaddcred</code> to create new credentials; it updates existing credentials. An alternative is <code>chkey</code> .
Directory object	A copy of the public key of each server that supports it.	Run the <code>/usr/lib/nis/nisupdkeys</code> command on the directory object.
Keyserver	The secret key of the NIS+ principal that is currently logged in.	Run <code>keylogin</code> for a principal user or <code>keylogin -r</code> for a principal workstation.
NIS+ daemon	Copies of directory objects, which in turn contain copies of their servers' public keys.	Kill the daemon and the cache manager. Then restart both.
Directory cache	A copy of directory objects, which in turn contain copies of their servers' public keys.	Kill the NIS+ cache manager and restart it with the <code>nis_cachemgr -i</code> command. The <code>-i</code> option resets the directory cache from the cold-start file and restarts the cache manager.
Cold-start file	A copy of a directory object, which in turn contains copies of its servers' public keys.	On the root master, kill the NIS+ daemon and restart it. The daemon reloads new information into the existing <code>NIS_COLD_START</code> file. For a client, first remove the cold-start and shared directory files from <code>/var/nis</code> , and kill the cache manager. Then re-initialize the principal with <code>nisinit -c</code> . The principal's trusted server reloads new information into the principal's existing cold-start file.

Table A-1 Where Credential-Related Information is Stored (Continued)

Item	Stores	To Reset or Change
passwd table	A user's password or a workstation's superuser password.	Use the <code>passwd -r nisplus</code> command. It changes the password in the NIS+ passwd table and updates it in the cred table.
passwd file	A user's password or a workstation's superuser password.	Use the <code>passwd -r nisplus</code> command, whether logged in as superuser or as yourself, whichever is appropriate.
passwd map (NIS)	A user's password or a workstation's superuser password.	Use <code>passwd -r nisplus</code> .

Updating Stale Cached Keys

The most commonly encountered out-of-date information is the existence of stale objects with old versions of a server's public key. You can usually correct this problem by running `nisupdkeys` on the domain you are trying to access. (See Chapter 5, "Administering NIS+ Credentials," for information on using the `nisupdkeys` command.)

Because some keys are stored in files or caches, `nisupdkeys` cannot always correct the problem. At times you might need to update the keys manually. To do that, you must understand how a server's public key, once created, is propagated through namespace objects. The process usually has five stages of propagation:

- "Stage 1: Server's Public Key Is Generated"
- "Stage 2: Public Key Is Propagated to Directory Objects"
- "Stage 3: Directory Objects Are Propagated Into Client Files"
- "Stage 4: When a Replica is Added to the Domain"
- "Stage 5: When the Server's Public Key Is Changed"

Each stage is described below.

Stage 1: Server's Public Key Is Generated

An NIS+ server is first an NIS+ client. So, its public key is generated in the same way as any other NIS+ client's public key: with the `nisaddcred` command. The public key is then stored in the cred table of the server's home domain, not of the domain the server will eventually support.

Stage 2: Public Key Is Propagated to Directory Objects

Once you have set up an NIS+ domain and an NIS+ server, you can associate the server with a domain. This association is performed by the `nismkdir` command. When the `nismkdir` command associates the server with the directory, it also copies the server's public key from the cred table to the domain's directory object. For example, assume the server is a client of the `wiz.com.` root domain, and is made the master server of the `sales.wiz.com.` domain.

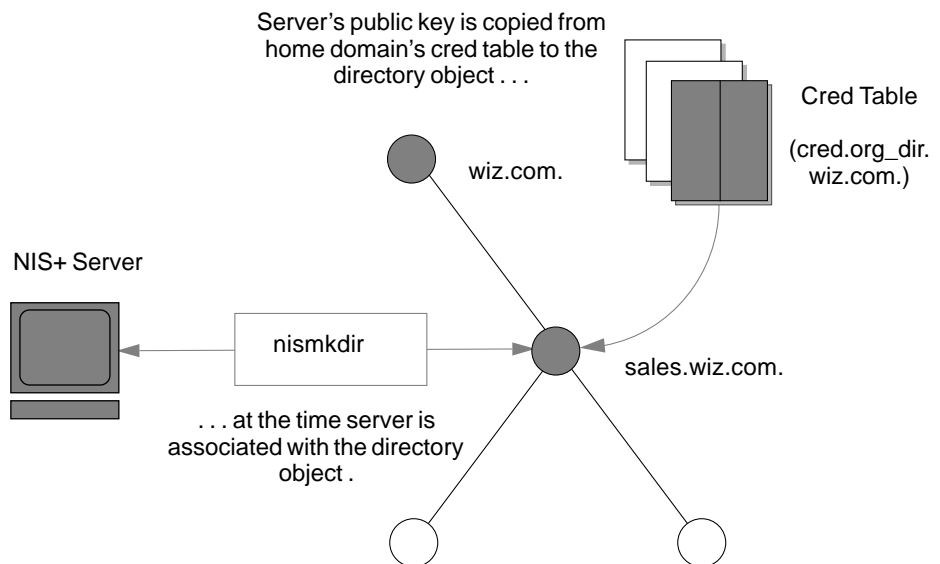


Figure A-1 Public Key is Propagated to Directory Objects

Its public key is copied from the `cred.org_dir.wiz.com.` domain and placed in the `sales.wiz.com.` directory object. This can be verified with the `niscat -o Sales.wiz.com.` command.

Stage 3: Directory Objects Are Propagated Into Client Files

All NIS+ clients are initialized with the `nisinit` utility or with the `nisclient` script.

Among other things, `nisinit` (or `nisclient`) creates a cold-start file `/var/nis/NIS_COLDSTART`. The cold-start file is used to initialize the client's directory cache `/var/nis/NIS_SHARED_DIRCACHE`. The cold-start file contains a copy of the directory object of the client's domain. Since the directory object already contains a copy of the server's public key, the key is now propagated into the cold-start file of the client.

In addition when a client makes a request to a server outside its home domain, a copy of the remote domains directory object is stored in the client's `NIS_SHARED_DIRCACHE` file. You can examine the contents of the client's cache by using the `nisshowcache` command, described on page 184.

This is the extent of the propagation until a replica is added to the domain or the server's key changes.

Stage 4: When a Replica is Added to the Domain

When a replica server is added to a domain, the `nisping` command (described on page 185) is used to download the NIS+ tables, including the cred table, to the new replica. Therefore, the original server's public key is now also stored in the replica server's cred table.

Stage 5: When the Server's Public Key Is Changed

If you decide to change DES credentials for the server (that is, for the root identity on the server), its public key will change. As a result, the public key stored for that server in the cred table will be different from those stored in:

- The cred table of replica servers (for a few minutes only)
- The main directory object of the domain supported by the server (until its time-to-live expires)
- The `NIS_COLDSTART` and `NIS_SHARED_DIRCACHE` files of every client of the domain supported by server (until their time-to-live expires, usually 12 hours)
- The `NIS_SHARED_DIRCACHE` file of clients who have made requests to the domain supported by the server (until their time-to-live expires)

Most of these locations will be updated automatically within a time ranging from a few minutes to 12 hours. To update the server's keys in these locations immediately, use the commands:

Table A-2 Updating a Server's Keys

Location	Command	See
cred table of replica servers (instead of using <code>nisping</code> , you can wait a few minutes until the table is updated automatically)	<code>nisping</code>	page 198
Directory object of domain supported by server	<code>nisupdkeys</code>	page 105
NIS_COLDSTART file of clients	<code>nisinit -c</code>	page 194
NIS_SHARED_DIRCACHE file of clients	<code>nis_cachemgr</code>	page 196

Note – You must first kill the existing `nis_cachemgr` before restarting `nis_cachemgr`.

Corrupted Credentials

When a principal (user or machine) has a corrupt credential, that principal is unable to perform any namespace operations or tasks, not even `nisls`. This is because a corrupt credential provides no permissions at all, not even the permissions granted to the nobody class.

Symptoms

User or root cannot perform any namespace tasks or operations. All namespace operations fail with a “permission denied” type of error message. The user or root cannot even perform a `nisls` operation.

Possible Cause

Corrupted keys or a corrupt, out-of-date, or otherwise incorrect `/etc/.rootkey` file.

Diagnosis

Use `snoop` to identify the bad credential.

Or, if the principal is listed, log in as the principal and try to run an NIS+ command on an object for which you are sure that the principal has proper authorization. For example, in most cases an object grants read authorization to the nobody class. Thus, the `nislsls object` command should work for any principal listed in the cred table. If the command fails with a “permission denied” error, then the principal’s credential is likely corrupted.

Solution

- *Ordinary user.* Perform a `keylogout` and then a `keylogin` for that principal.
- *Root/superuser.* Run `keylogout -f` followed by `keylogin -r`.

Keysevr *Failure*

The `keysevr` is unable to encrypt a session. There are several possible causes for this type of problem:

Possible Causes and Solutions

- The client has not keylogged in. Make sure that the client is keylogged in. To determine if a client is properly keylogged in, have the client run `nisdefaults -v` (or run it yourself as the client). If `(not authenticated)` is returned on the `Principal Name` line, the client is not properly keylogged in.
- The client (host) does not have appropriate LOCAL or DES credentials. Run `niscat` on the client’s cred table to verify that the client has appropriate credentials. If necessary, add credentials as explained in “Creating Credential Information for NIS+ Principals” on page 91.
- The `keysevr` daemon is not running. Use the `ps` command to see if `keysevr` is running. If it is not running, restart it and then do a `keylogin`.
- While `keysevr` is running, other long running processes that make secure RPC or NIS+ calls are not. For example, `automountd`, `rpc.nisd`, and `sendmail`. Verify that these processes are running correctly. If they are not, restart them.

Machine Previously Was an NIS+ Client

If this machine has been initialized before as an NIS+ client of the same domain, try `keylogin -r` and enter the root login password at the Secure RPC password prompt.

No Entry in the cred Table

To make sure that an NIS+ password for the principal (user or host) exists in the cred table, run the following command in the principal's home domain

```
nisgrep -A cname=principal cred.org_dir.domainname
```

If you are running `nisgrep` from another domain, the *domainname* must be fully qualified.

Changed Domain Name

Do not change a domain name.

If you change the name of an existing domain you will create authentication problems because the fully qualified original domain name is embedded in objects throughout your network.

If you have *already* changed a domain name and are experiencing authentication problems, or error messages containing terms like “malformed” or “illegal” in relation to a domain name, change the domain name back to its original name. The recommended procedure for renaming your domains is to create a *new* domain with the *new* name, set up your machines as servers and clients of the new domain, make sure they are performing correctly, and then remove the old domain.

When Changing a Machine to a Different Domain

If this machine is an NIS+ client and you are trying to change it to a client of a different domain, remove the `/etc/.rootkey` file, and then rerun the `nisclient` script using the network password supplied by your network administrator or taken from the `nispopulate` script.

NIS+ Password and Login Password in /etc/passwd File

Your NIS+ password is stored in the NIS+ passwd table. Your user login password may be stored in NIS+ passwd table or in your `/etc/passwd` file. (Your user password and NIS+ password can be the same or different.) To change a password in an `/etc/passwd` file, you must run the `passwd` command with the `nsswitch.conf` file set to `files` or with the `-r files` flag.

The `nsswitch.conf` file specifies which password is used for which purpose. If the `nsswitch.conf` file is directing system queries to the wrong location, you will get password and permission errors.

Secure RPC Password and Login Passwords Are Different

When a principal's login password is different from his or her Secure RPC password, `keylogin` cannot decrypt it at login time because `keylogin` defaults to using the principal's login password, and the private key was encrypted using the principal's Secure RPC password.

When this occurs the principal can log in to the system, but for NIS+ purposes is placed in the authorization class of `nobody` because the keyserver does not have a decrypted private key for that user. Since most NIS+ environments are set up to deny the `nobody` class create, destroy, and modify rights to most NIS+ objects this results in "permission denied" types errors when the user tries to access NIS+ objects.

In this context **network password** is sometimes used as a synonym for Secure RPC password. When prompted for your network password, enter your Secure RPC password.

To be placed in one of the other authorization classes, a user in this situation must explicitly run the `keylogin` program and give the principal's Secure RPC password when `keylogin` prompts for password. (See "Keylogin" on page 99.)

But an explicit `keylogin` provides only a temporary solution that is good only for the current login session. The keyserver now has a decrypted private key for the user, but the private key in the user's cred table is still encrypted using the user's Secure RPC password, which is different than the user's login password. The next time the user logs in, the same problem reoccurs. To permanently solve the problem the user needs to change the private key in the cred table to one based on the user's login ID rather than the user's Secure RPC password. To do this, the user need to run the `chkey` program as described in "Changing Keys for a NIS+ Principal" on page 100.

Thus, to permanently solve a Secure RPC password different than login password problems, the user (or an administrator acting for the user) must perform the following steps:

1. Log in using the login password.
2. Run the `keylogin` program to temporarily get a decrypted private key stored in the keyserver and thus gain temporary NIS+ access privileges.
3. Run `chkey -p` to permanently change the encrypted private key in the cred table to one based on the user's login password.

Preexisting /etc/.rootkey File

Symptoms

Various “insufficient permission to” and “permission denied” error messages.

Possible Cause

An `/etc/.rootkey` file already existed when you set up or initialized a server or client. This could occur if NIS+ had been previously installed on the machine and the `.rootkey` file was not erased when NIS+ was removed or the machine returned to using NIS or `/etc` files.

Diagnosis

Run `ls -l` on the `/etc` directory and `nislsls -l org_dir` and compare the date of the `/etc/.rootkey` to the date of the cred table. If the `/etc/.rootkey` date is clearly earlier than that of the cred table, it may be a preexisting file.

Solution

Run `keylogin -r` as root on the problem machine and then set up the machine as a client again.

Root Password Change Causes Problem

Symptoms

You change the root password on a machine, and the change either fails to take effect or you are unable to log in as superuser.

Possible Cause

You should not have UserID 0 in the passwd table.

You changed the root password, but root's key was not properly updated. Either because you forgot to run `chkey -p` for root or some problem came up.

Solution

Log in as a user with administration privileges (that is, a user who is a member of a group with administration privileges) and use `passwd` to restore the old password. Make sure that old password works. Now use `passwd` to change root's password to the new one, and then run `chkey -p`.



Caution – Once your NIS+ namespace is set up and running, you can change the root password on the root master machine. But do not change the root master keys, as these are embedded in all directory objects on all clients, replicas, and servers of subdomains. To avoid changing the root master keys, always use the `-p` option when running `chkey` as root.

Slow Performance and System Hang Problems

This section describes common slow performance and system hang problems.

Symptoms

Error messages with operative clauses such as:

- “Busy try again later”
- “Not responding”

Other common symptoms:

- You issue a command and nothing seems to happen for far too long.
- Your system, or shell, no longer responds to keyboard or mouse commands.

- NIS+ operations seem to run slower than they should or slower than they did before.

Checkpointing

Someone has issued an `nisping` or `nisping -C` command. Or the `rpc.nisd` daemon is performing a checkpoint operation.



Caution – Do not reboot! Do not issue any more `nisping` commands.

When performing a `nisping` or checkpoint, the server will be sluggish or may not immediately respond to other commands. Depending on the size of your namespace, these commands may take a noticeable amount of time to complete. Delays caused by checkpoint or ping commands are multiplied if you, or someone else, enter several such commands at one time. Do not reboot. This kind of problem will solve itself. Just wait until the server finishes performing the `nisping` or checkpoint command.

During a full master-replica resync, the involved replica server will be taken out of service until the resync is complete. Do not reboot—just wait.

Variable `NIS_PATH`

Make sure that your `NIS_PATH` variable is set to something clean and simple. For example, the default: `org_dir.$:$`. A complex `NIS_PATH`, particularly one that itself contains a variable, will slow your system and may cause some operations to fail. (See “`NIS_PATH` Environment Variable” on page 43 for more information.)

Do not use `nistbladm` to set nondefault table paths. Nondefault table paths will slow performance.

Table Paths

Do not use table paths because they will slow performance.

Too Many Replicas

Too many replicas for a domain degrade system performance during replication. There should be no more than 10 replicas in a given domain or subdomain. If you have more than five replicas in a domain, try removing some of them to see if that improves performance.

Recursive Groups

A recursive group is a group that contains the name of some other group. While including other groups in a group reduces your work as system administrator, doing so slows down the system. You should not use recursive groups.

Large NIS+ Database Logs at Start-up

When `rpc.nisd` starts up it goes through each log. If the logs are long, this process could take a long time. If your logs are long, you may want to checkpoint them using `nisping -C` before starting `rpc.nisd`.

The Master `rpc.nisd` Daemon Died

Symptoms

If you used the `-M` option to specify that your request be sent to the master server, and the `rpc.nisd` daemon has died on that machine, you will get a “server not responding” type error message and no updates will be permitted. (If you did not use the `-M` option, your request will be automatically routed to a functioning replica server.)

Possible Cause

Using uppercase letters in the name of a home directory or host can sometimes cause `rpc.nisd` to die.

Diagnosis

First make sure that the server itself is up and running. If it is, run `ps -ef | grep rpc.nisd` to see if the daemon is still running.

Solution

If the daemon has died, restart it. If `rpc.nisd` frequently dies, contact your service provider.

No `nis_cachemgr`**Symptoms**

It takes too long for a machine to locate namespace objects in other domains.

Possible Cause

You do not have `nis_cachemgr` running.

Diagnosis

Run `ps -ef | grep nis_cachemgr` to see if it is still running.

Solution

Start `nis_cachemgr` on that machine.

Server Very Slow at Start-up After NIS+ Installation**Symptoms**

A server performs slowly and sluggishly after using the NIS+ scripts to install NIS+ on it.

Possible Cause

You forgot to run `nisping -C -a` after running the `nispopulate` script.

Solution

Run `nisping -C -a` to checkpoint the system as soon as you are able to do so.

niscat *Returns:* Server busy. Try Again

Symptoms

You run `niscat` and get an error message indicating that the server is busy.

Possible Cause

- The server is busy with a heavy load, such as when doing a resync.
- The server is out of swap space.

Diagnosis

Run `swap -s` to check your server's swap space.

Solution

You must have adequate swap and disk space to run NIS+. If necessary, increase your space.

NIS+ Queries Hang After Changing Host Name

Symptoms

Setting the host name for an NIS+ server to be fully qualified is not recommended. If you do so, and NIS+ queries then just hang with no error messages, check the following possibilities:

Possible Cause

Fully qualified host names must meet the following criteria:

- The domain part of the host name must be the same as the name returned by the `domainname` command.
- After the setting the host name to be fully qualified, you must also update all the necessary `/etc` and `/etc/inet` files with the new host name information.
- The host name must end in a period.

If you started `rpc.nisd` with the `-B` option, you must also kill the `rpc.nisd_resolv` daemon.

Solution

Kill the NIS+ processes that are hanging and then kill `rpc.nisd` on that host or server. Rename the host to match the two requirements listed above. Then reinitialize the server with `nisinit`. (If queries still hang after you are sure that the host is correctly named, check other problem possibilities in this section.)

System Resource Problems

This section describes problems having to do with lack of system resources such as memory, disk space, and so forth.

Symptoms

Error messages with operative clauses such as:

- “No memory”
- “Out of disk space”
- “Cannot [do something] with log”
- “Unable to fork”

Insufficient Memory

Lack of sufficient memory or swap space on the system you are working with will cause a wide variety of NIS+ problems and error messages. As a short-term, temporary solution, try to free additional memory by killing unneeded windows and processes. If necessary, exit your windowing system and work from the terminal command line. If you still get messages indicating inadequate memory, you will have to install additional swap space or memory, or switch to a different system that has enough swap space or memory.

Under some circumstances, applications and processes may develop memory leaks and grow too large. you can check the current size of an application or process by running:

```
ps -el
```

The `sz` (size) column shows the current memory size of each process. If necessary, compare the sizes with comparable processes and applications on a machine that is not having memory problems to see if any have grown too large.

Insufficient Disk Space

Lack of disk space will cause a variety of error messages. A common cause of insufficient disk space is failure to regularly remove `tmp` files and truncate log files. Log and `tmp` files grow steadily larger unless truncated. The speed at which these files grow varies from system to system and with the system state. Log files on a system that is working inefficiently or having namespace problems will grow very fast.

Note – If you are doing a lot of troubleshooting, check your log and `/tmp` files frequently. Truncate log files and remove `/tmp` files before lack of disk space creates additional problems. Also check the root directory and home directories for core files and delete them.

The way to truncate log files is to regularly checkpoint your system (Keep in mind that a checkpoint process may take some time and will slow down your system while it is being performed, checkpointing also requires enough disk space to create a complete copy of the files before they are truncated.)

To checkpoint a system, run `nisping -C`.

Insufficient Processes

On a heavily loaded machine it is possible that you could reach the maximum number of simultaneous processes that the machine is configured to handle. This causes messages with clauses like “unable to fork”. The recommended method of handling this problem is to kill any unnecessary processes. If the problem persists, you can reconfigure the machine to handle more processes as described in your system administration documentation.

User Problems

This section describes NIS+ problems that a typical user might encounter.

Symptoms

- User cannot log in.
- User cannot `rlogin` to other domain

User Cannot Log In

There are many possible reasons for a user being unable to log in:

- *User forgot password.* To set up a new password for a user who has forgotten the previous one, run `nispaswd` for that user on another machine (naturally, you have to be the NIS+ administrator to do this).
- *Mistyping password.* Make sure the user knows the correct password and understands that passwords are case-sensitive and that the letter “o” is not interchangeable with the numeral “0,” nor is the letter “l” the same as the numeral “1.”
- *“Login incorrect” type message.* For causes other than simply mistyping the password, see ““Login Incorrect” Message” on page 321.
- The user’s password privileges have expired (see “Password Privilege Expiration” on page 164).
- An inactivity maximum has been set for this user, and the user has passed it (see “Specifying Maximum Number of Inactive Days” on page 166).
- The user’s `nsswitch.conf` file is incorrect. The `passwd` entry in that file must be one of the following five permitted configurations:
 - `passwd: files`
 - `passwd: files nis`
 - `passwd: files nisplus`
 - `passwd: compat`
 - `passwd: compat`
`passwd_compat: nisplus`

Any other configuration will prevent a user from logging in.

(See “`nsswitch.conf` File Requirements” on page 144 for further details.)

User Cannot Log In Using New Password

Symptoms

Users who recently changed their password are unable to log in at all, or are able to log in on some machines but not on others.

Possible Causes

- It may take some time for the new password to propagate through the network. Have users try to log in with the old password.
- The password was changed on a machine that was not running NIS+ (see “User Cannot Log In Using New Password” on page 340), or `nispasswd` was not used to create the new password.

User Cannot Remote Log In to Remote Domain

Symptoms

User tries to `rlogin` to a machine in some other domain and is refused with a “Permission denied” type error message.

Possible Cause

To `rlogin` to a machine in another domain, a user must have LOCAL credentials in that domain.

Diagnosis

Run `nismatch username.domainname.cred.org_dir` in the other domain to see if the user has a LOCAL credential in that domain.

Solution

Go to the remote domain and use `nisaddcred` to create a LOCAL credential for the user in the that domain.

User Cannot Change Password

The most common cause of a user being unable to change passwords is that the user is mistyping (or has forgotten) the old password.

Other possible causes:

- The password Min value has been set to be greater than the password Max value. See page 161.
- The password is locked or expired. See ““Login Incorrect” Message” on page 321 and “Password Locked, Expired, or Terminated” on page 322.

Other NIS+ Problems

This section describes problems that do not fit any of the previous categories.

How to Tell if NIS+ Is Running

You may need to know whether a given host is running NIS+. A script may also need to determine whether NIS+ is running.

You can assume that NIS+ is running if:

- `nis_cachemgr` is running.
- The host has a `/var/nis/NIS_COLD_START` file.
- `nislsls` succeeds.

Replica Update Failure

Symptoms

Error messages indicating that the update was not successfully complete. (Note that the message: `replica_update: number` updates `number` errors indicates a successful update.)

Possible Causes

Any of the following error messages indicate that the server was busy and that the update should be rescheduled:

- `Master server busy, full dump rescheduled`
- `replica_update: error result was Master server busy, full dump rescheduled`
- `replica_update: master server busy, rescheduling the resync`

- `replica_update: master server busy, will try later`
- `replica_update: nis dump result Master server busy, full dump rescheduled`
- `nis_dump_svc: one replica is already resynching`

(These messages are generated by, or in conjunction with, the NIS+ error code constant: `NIS_DUMPLATER`.)

These messages indicate that there was some other problem:

- `replica_update: error result was ...`
- `replica_update: nis dump result nis_perror error string`
- `root_replica_update: update failed string-variable: could not fetch object from master`

(If `rpc.nisd` is being run with the `-C` (open diagnostic channel) option, additional information may be entered in either the master server or replica server's system log.)

These messages indicate possible problems such as:

- The server is out of child processes that can be allocated.
- A read-only child process was requested to dump.
- Another replica is currently resynching.

Diagnosis

Check both the replica and server's system log for additional information. How much, if any, additional information is recorded in the system logs depends on your system's error reporting level, and whether or not you are running `rpc.nisd` with the `-C` option (diagnostics).

Solution

In most cases, these messages indicate minor software problems which the system is capable of correcting. If the message was the result of a command, simply wait for a while and then try the command again. If these messages appear often, you can change the threshold level in your `/etc/syslog.conf` file. See the `syslog.conf` man page for details.

FNS Problems and Solutions

This section presents problem scenarios with a description of probable causes, diagnoses, and solutions.

Cannot Obtain Initial Context

Symptom

I get the message “Cannot obtain initial context.”

Possible Cause

This is caused by an installation problem.

Diagnosis

Check that FNS has been installed properly by looking for the file, `/usr/lib/fn/fn_ctx_initial.so`.

Solution

Install the `fn_ctx_initial.so` library.

Nothing in Initial Context

Symptom

I run `fnlist` to look at what is in the initial context but see nothing.

Possible Cause

This is caused by an NIS+ configuration problem. The organization associated with the user and machine running the `fn*` commands do not have an associated `ctx_dir` directory.

Diagnosis

Use the `nislsl` command to see whether there is a `ctx_dir` directory.

Solution

If there is no `ctx_dir` directory, run `fncreate -t org/nis+_domain_name/` to create the `ctx_dir` directory.

“No Permission” Messages (FNS)**Symptom**

I get “no permission” messages.

Possible Cause

“No permission” messages mean that you do not have access to perform the command.

Diagnosis

Check permission using the appropriate NIS+ commands, described in “Advanced FNS and NIS+ Issues” on page 280. Use the `nisdefaults` command to determine your NIS+ principal name.

Another area to check is whether you are using the right name. For example, `org//` names the context of the root organization. Make sure you have permission to manipulate the root organization. Or maybe you meant to specify `myorgunit/`, instead.

Solution

If you do have permission, then the appropriate credentials probably have not been acquired.

This could be caused by the following:

- a `keylogin` has not been performed (defaults to NIS+ principal “nobody”)
- a `keylogin` was made to a source other than NIS+
 - Check that the `/etc/nsswitch.conf` file has a `publickey: nisplus` entry.
 - This might manifest itself as an authentication error.

`fnlist` *Does not List Suborganizations*

Symptom

I run `fnlist` with an organization name, expecting to see suborganizations, but instead see nothing.

Possible Cause

This is caused by an NIS+ configuration problem. Suborganizations must be NIS+ domains. By definition, an NIS+ domain must have a subdirectory named `org_dir`.

Diagnosis

Use the `nisls` command to see what subdirectories exist. Run `nisls` on each subdirectory to verify which subdirectories have an `org_dir`. The subdirectories with an `org_dir` are suborganizations.

Solution

Not applicable.

Cannot Create Host- or User-related Contexts

Symptom

When I run `fncreate -t` for the `user`, `username`, `host`, or `hostname` contexts, nothing happens.

Possible Cause

You have not set the `NIS_GROUP` environment variable. When you create a `user` or `host` context it is owned by the `host` or `user`, and not by the administrator who set up the namespace. Therefore, `fncreate` requires that the `NIS_GROUP` variable be set to enable the administrators part of that group to manipulate the contexts.

Diagnosis

Check the `NIS_GROUP` environment variable.

Solution

The `NIS_GROUP` environment variable should be set to the group name of the administrators who will administer the contexts.

Cannot Remove a Context I Created**Symptom**

When I run `fndestroy` on a host or user context the context is not removed.

Possible Cause

You do not own the host or user context. When you create a user or host context it is owned by the host or user, and not by the administrator who set up the namespace.

Diagnosis

Check the `NIS_GROUP` environment variable.

Solution

The `NIS_GROUP` environment variable needs to be set to the group name of the administrator who will administer the contexts.

“Name in Use” with fnunbind**Symptom**

I get “name in use” when trying to remove bindings. It works for certain names but not for others.

Possible Cause

You cannot unbind the name of a context. This restriction is in place to avoid leaving behind contexts that have no name (“orphaned contexts”).

Diagnosis

Run the `fnlist` command on the name to verify that it is a context.

Solution

If the name is a context, use the `fndestroy` command to destroy the context.

“Name in Use” with `fnbind/fncreate -s`**Symptom**

I use the `-s` option with `fnbind` and `fncreate`, but for certain names I get “name in use.”

Possible Cause

`fnbind -s` and `fncreate -s` overwrite the existing binding if it already exists; but if the old binding is one that must be kept to avoid orphaned contexts, the operation fails with a “name in use” error because the binding could not be removed. This is done to avoid orphaned contexts.

Diagnosis

Run the `fnlist` command on the name to verify that it is a context.

Solution

Run the `fndestroy` command to remove the context *before* running `fnbind` or `fncreate` on the same name.

`fndestroy/fnunbind` and “Operation Failed”**Symptom**

When I do an `fndestroy` or `fnunbind` on certain names that I know do *not* exist, I receive no indication that the operation failed.

Possible Cause

The operation did not fail. The semantics of `fndestroy` and `fnunbind` are that if the terminal name is not bound, the operation returns success.



Diagnosis

Run the `fnlookup` command on the name. You should receive the message, “name not found.”

Solution

Not applicable.

Error Messages



This section alphabetically lists the more common NIS+ error messages. For each message there is an explanation and, where appropriate, a solution or a cross-reference to some other portion of this manual.

Appendix A, “Problems and Solutions,” describes various type of problems and their solutions. Where appropriate, error messages in this appendix are cross-referenced to the corresponding section in Appendix A.

About NIS+ and FNS Error Messages

Some of the error messages documented in this chapter are documented more fully in the appropriate man pages.

FNS messages are encapsulated in the `FN_status_t` object as status codes. See the `FN_status_t(3)` man page for the corresponding status codes

Error Message Context

Error messages may appear in pop-up windows, shell tool command lines, user console window, the `syslog` file, or in log files. You can raise or lower the severity threshold level for reporting error conditions in your `/etc/syslog.conf` file.

In the most cases, the error messages that you see are generated by the commands you issued or the container object (table or directory) your command is addressing. However, in some cases an error message may be

If you cannot trace the cause of an error message to your command or machine, consider the possibility that the message may have been generated by a server in response to your command or in response to some other NIS+ function.

generated by a server invoked in response to your command (these messages usually show in `syslog`). For example, a “permission denied” message most likely refers to you, or the machine you are using, but it could also be caused by software on a server not having the correct permissions to carry out some function passed on to it by your command or your machine.

Similarly, some commands cause a number of different NIS+ objects to be searched or queried. Some of these objects may not be obvious. Any one of these objects could return an error message regarding permissions, read-only state, unavailability, and so forth. In such cases the message may or may not be able to inform you of which object the problem occurred in.

In normal operation, the NIS+ software and servers make routine NIS+ function calls. Sometimes those calls fail and in doing so generate an error message. It occasionally happens that before a client or server processes your most recent command, some other NIS+ call fails and you see the resulting error message. Such a message might appear as if it were in response to your command, when in fact it is in response to some other operation.

Note – When working with an NIS+ namespace you may encounter error messages generated by remote procedure calls. These RPC error messages are not documented here. Check your system documentation.

Context-Sensitive Meanings

A single NIS+ error message may have slightly different meanings depending on which part of the NIS+ software generated the message. For example, when a “Not Found” type message is generated by the `nislsls` command it means that there are no NIS+ objects that have the specified name, but when it is generated by the `nismatch` command it means that no table entries were found that meet the search criteria.

How Error Messages Are Alphabetized

Some error messages may be preceded by a character string (a name or a number) or by the name of the routine that generated the error message. In some cases a character string may also follow an error message. In this appendix, these variable strings are indicated by an *italic typeface*.

The error messages in this appendix are sorted alphabetically according to the following rules:

- Capitalization is ignored. Thus, messages that begin with “A” and “a” are alphabetized together.
- Nonalphabetic symbols are ignored. Thus, a message that begins with `_svcauth_des` is listed with the other messages that begin with the letter “S.”
- Many messages contain variable strings such as user IDs, domain names, host names, and so forth. Because variables could be anything, they are not included in the sorting of the messages listed in this appendix. For example, the actual message `Sales: is not a table` would be listed in this appendix as: `name: is not a table` and would be alphabetized under the letter “I” for the first nonvariable letter.
- Error messages that begin with asterisks, such as `**ERROR: domainname does not exist`, are generated by the NIS+ installation and setup scripts. They are alphabetized according to their first letter, ignoring the asterisks.

Some of the error messages documented in this chapter are documented more fully in the appropriate man pages.

Common NIS+ and FNS Error Messages

`abort_transaction: Failed to action NIS+ objectname`

The `abort_transaction` routine failed to back out of an incomplete transaction due to a server crash or some other unrecoverable error. See “Namespace Database Problems” on page 311 for further information.

`abort_transaction: Internal database error`

`abort_transaction: Internal error, log entry corrupt NIS+ objectname`

These two messages indicate some form of corruption in a namespace database or log. See “Namespace Database Problems” on page 311 for additional information.

add_cleanup: Cant allocate more rags.

This message indicates that your system is running low on available memory. See “Insufficient Memory” on page 337 for information on insufficient memory problems.

add_pingitem: Couldn't add *directoryname* to pinglist (no memory)

See “Insufficient Memory” on page 337 for information on low memory problems.

add_update: Attempt add transaction from read only child.
add_update Warning: attempt add transaction from read only child

An attempt by a read-only child `rpc.nisd` process to add an entry to a log. An occasional appearance of this message in a log is not serious. If this message appears frequently, contact the Sun Solutions Center.

Attempting to free a free rag!

This message indicates a software problem with `rpc.nisd`. The `rpc.nisd` should have aborted. Run `ps -ef | grep rpc.nisd` to see if `rpc.nisd` is still running. If it is, kill it and restart it with the same options as previously used. If it is not running, restart it with the same options as previously used. Check `/var/nis` to see if a `core` file has been dumped. If there is a `core` file, delete it.

Attempt to remove a non-empty table

An attempt has been made by `nistbladm` to remove an NIS+ table that still contains entries. Or by `nisrmdir` to remove a directory that contains files or subdirectories. If you are trying to delete a directory, use `nislsls -lR` to check for existing files or subdirectories and delete them first. If you are trying to delete a table, use `niscat` to check the contents of the table and `nistbladm` to delete any existing contents.

This message is generated by the NIS+ error code constant: `NIS_NOTEMPTY`. See the `nis_tables` man page for additional information.

attribute no permission

FNS error message. The caller did not have permission to perform the attempted attribute operation.

If you started `rpc.nisd` with the `-YB` option, you must also kill the `rpc.nisd_resolv` daemon.

attribute value required

FNS error message. The operation attempted to create an attribute without a value, and the specific naming system does not allow this.

authdes_marshall: DES encryption failure

DES encryption for some authentication data failed. Possible causes:

- Corruption of a library function or argument.
- A problem with a DES encryption chip, if you are using one.

Call the Sun Solutions Center for assistance.

authdes_refresh: keyserv is unable to encrypt session key

The `keyserv` process was unable to encrypt the session key with the public key that it was given. See “Keyserv Failure” on page 328 for additional information.

authdes_refresh: unable to encrypt conversation key

The `keyserv` process could not encrypt the session key with the public key that was given. This usually requires some action on your part. Possible causes are:

- The `keyserv` process is dead or not responding. Use `ps -ef` to check whether the `keyserv` process is running on the `keyserv` host. If it is not, then start it, and then run `keylogin`.
- The client has not performed a `keylogin`. Do a `keylogin` for the client and see if that corrects the problem.
- The client host does not have credentials. Run `nismatch` on the client’s home domain cred table to see if the client host has the proper credentials. If it does not, create them.
- A DES encryption failure. See the `authdes_marshall: DES encryption failure` error message).

See “Security Problems” on page 321 for additional information regarding security key problems.

`authdes_refresh: unable to synchronize clock`

This indicates a synchronization failure between client and server clocks. This will usually correct itself. However, if this message is followed by any time stamp related error, you should manually resynchronize the clocks. If the problem reoccurs, check that remote `rpcbind` is functioning correctly.

`authdes_refresh: unable to synch up w/server`

The client-server clock synchronization has failed. This could be caused by the `rpcbind` process on the server not responding. Use `ps -ef` on the server to see if `rpcbind` is running. If it is not, restart it. If this error message is followed by any time stamp-related message, then you need use `rdate servername` to manually resync the client clock to the server clock.

`authdes_seccreate: key serv is unable to generate session key`

This indicates that `key serv` was unable to generate a random DES key for this session. This requires some action on your part:

- Check to make sure that `key serv` is running properly. If it is not, restart it along with all other long-running processes that use Secure RPC or make NIS+ calls such as `automountd`, `rpc.nisd` and `sendmail`. Then do a `keylogin`.
- If `key serv` is up and running properly, restart the process that logged this error.

`authdes_seccreate: no public key found for servername`

The client side cannot get a DES credential for the server named *servername*. This requires some action on your part:

- Check to make sure that *servername* has DES credentials. If it does not, create them.
- Check the switch configuration file to see which name service is specified and then make sure that service is responding. If it is not responding, restart it.

`authdes_seccreate: out of memory`

See “System Resource Problems” on page 337 for information on insufficient memory problems.

`authdes_seccreate: unable to gen conversation key`

The `keyserv` process was unable to generate a random DES key. The most likely cause is that the `keyserv` process is down or otherwise not responding. Use `ps -ef` to check whether the `keyserv` process is running on the `keyserv` host. If it is not, then start it and then run `keylogin`.

If restarting `keyserv` fails to correct the problem, it may be that other processes that use Secure RPC or make NIS+ calls are not running (for example, `automountd`, `rpc.nisd`, or `sendmail`). Check to see whether these processes are running, if they are not, restart them.

See “Security Problems” on page 321 for additional information regarding security key problems.

`authdes_validate: DES decryption failure`

See `authdes_marshal: DES decryption failure` on page 353.

`authdes_validate: verifier mismatch`

The time stamp that the client sent to the server does not match the one received from the server. (This is not recoverable within a Secure RPC session. Possible causes:

- Corruption of the session key or time stamp data in the client or server cache
- The server deleted from this cache a session key for a still active session.
- Network data corruption.

Try re-executing the command.

`authentication failure`

FNS error message. The operation could not be completed because the principal making the request cannot be authenticated with the name service involved. If the service is NIS+, check that you are identified as the correct principal (run the command `nisdefaults`) and that your machine has specified the correct source for `publickeys`. Check that the `/etc/nsswitch.conf` file has the entry, `publickey: nisplus`.

bad reference

FNS error message. FNS could not interpret the contents of the reference. This may result if the contents of the reference has been corrupted or when the reference identifies itself as an FNS reference, but FNS doesn't know how to decode it.

CacheBind: xdr_directory_obj failed.

The most likely causes for this message are:

- Bad or incorrect parameters being passed to the `xdr_directory_obj` routine. Check the syntax and accuracy of whatever command you most recently entered.
- An attempt to allocate system memory failed. See “Insufficient Memory” on page 337 for a discussion of memory problems.
- If your command syntax is correct, and your system does not seem to be short of memory, contact the Sun Solutions Center.

Cache expired

The entry returned came from an object cache that has expired. This means that the time-to-live value has gone to zero and the entry may have changed. If the flag `NO_CACHE` was passed to the lookup function, then the lookup function will retry the operation to get an unexpired copy of the object.

This message is generated by the NIS+ error code constant: `NIS_CACHEEXPIRED`. See the `nis_tables` and `nis_names` man pages for additional information.

Callback: - select failed *message number*

An internal system call failed. In most cases this problem will correct itself. If it does not correct itself, make sure that `rpc.nisd` has not been aborted. If it has, restart it. If the problem reoccurs frequently, contact the Sun Solutions Center.

CALLBACK_SVC: bad argument

An internal system call failed. In most cases this problem will correct itself. If it does not correct itself, make sure that `rpc.nisd` has not been aborted. If it has, restart it. If the problem reoccurs frequently, contact the Sun Solutions Center.

Cannot grow transaction log error *string-variable*

The system cannot add to the log file. The reason is indicated by the *string-variable*. The most common cause of this message is lack of disk space. See “Insufficient Disk Space” on page 338.

Cannot obtain Initial Context

FNS error message. Indicates an installation problem. See “Cannot Obtain Initial Context” on page 343.

Cannot truncate transaction log file

An attempt has been made to checkpoint the log, and the `rpc.nisd` daemon is trying to shrink the log file after deleting the checkpointed entries from the log. See the `ftruncate` man pages for a description of various factors that might cause this routine to fail. See also “Namespace Database Problems” on page 311.

Cannot write one character to transaction log, error *message*

An attempt has been made by the `rpc.nisd` daemon to add an update from the current transaction into the transaction log, and the attempt has failed for the reason given in the *message* that has been returned by the function. Additional information may be obtained from the `write` routine’s man page.

Can’t compile regular expression *variable*

Returned by the `nisgrep` command when the expression in `keypat` was malformed.

Can’t find name service for `passwd`

Either there is no `nsswitch.conf` file or there is no `passwd` entry in the file, or the `passwd` entry does not make sense or is not one of the allowed formats.

Can’t find *name*’s secret key

Possible causes:

- You may have incorrectly types the password.
- There may be no entry for *name* in the cred table.

- NIS+ could not decrypt the key (possibly because the entry might be corrupt).
- The `nsswitch.conf` file may be directing the query to a local password in an `/etc/passwd` file that is different than the NIS+ password recorded in the cred table.

See “Security Problems” on page 321 for information on diagnosing and solving these type of problem.

`checkpoint_log`: Called from read only child ignored.

This is simply a status message indicating that a read-only process attempted to perform an operation restricted to the parent process, and the attempt was aborted. No action need be taken.

`checkpoint_log`: Unable to checkpoint, log unstable.

An attempt was made to checkpoint a log that was not in a stable state. (That is, the log was in a resync, update, or checkpoint state.) Wait until the log is stable, and then rerun the `nisping` command.

`check_updaters`: Starting resync.

This is simply a system status message. No action need be taken.

Child process requested to checkpoint!

This message indicates a minor software problem that the system is capable of correcting. If these messages appear often, you can change the threshold level in your `/etc/syslog.conf` file. See the `syslog.conf` man page for details.

Column not found: *columnname*

The specified column does not exist in the specified table.

communication failure

FNS error message. FNS could not communicate with the name service to complete the operation.

configuration error

An error resulted because of configuration problems. Examples:

- (1) the bindings table are removed out-of-band (outside of FNS).

(2) a host is in the NIS+ hosts directory object but does not have a corresponding FNS host context.

context not empty

FNS error message. An attempt has been made to remove a context that still contains bindings.

continue operation using status values

FNS error message. The operation should be continued using the remaining name and the resolved reference returned in the status.

Could not find *string-variable*'s secret key

Possible causes:

- You may have incorrectly typed the password.
- There may be no entry for name in the cred table.
- NIS+ could not decrypt the key (possibly because the entry might be corrupt)
- The `nsswitch.conf` file may have the wrong publickey policy. It may be directing the query to a local public key in an `/etc/publickey` file that is different from the NIS+ password recorded in the cred table.

See “Security Problems” on page 321 for information on diagnosing and solving these types of problem.

Could not generate netname

The Secure RPC software could not generate the Secure RPC netname for your UID when performing a `keylogin`. This could be due to the following causes:

- You do not have LOCAL credentials in the NIS+ cred table of the machine's home domain.
- You have a local entry in `/etc/passwd` with a UID that is different from the UID you have in the NIS+ passwd table.

String-variable: could not get secret key for '*string-variable*'

Possible causes:

- You may have incorrectly typed the password.
- There may be no entry for name in the cred table.

- NIS+ could not decrypt the key (possibly because the entry might be corrupt)
- The `nsswitch.conf` file may have the wrong `publickey` policy. It may be directing the query to a local `publickey` in an `/etc/publickey` file that is different from the NIS+ password recorded in the `cred` table.

See “Security Problems” on page 321 for information on diagnosing and solving these type of problem.

Couldn't fork a process!

The server could not fork a child process to satisfy a callback request. This is probably caused by your system reaching its maximum number of processes. You can kill some unneeded processes, or increase the number of processes your system can handle. See “Insufficient Processes” on page 338 for additional information.

Couldn't parse access rights for column *string-variable*

This message is usually returned by the `nistbladm -u` command when something other than a + (plus sign), a - (minus sign), or an = (equal sign) is entered as the operator. Other possible causes are failure to separate different column rights with a comma, or the entry of something other than `r`, `d`, `c`, or `m` for the type of permission. Check the syntax for this type of entry error. If everything is entered correctly and you still get this error, the table might have been corrupted.

Database for table does not exist

Attempt to look up a table has failed. See “Object Not Found Problems” on page 314 for possible causes.

This message is generated by the NIS+ error code constant: `NIS_NOSUCHTABLE`. See the `nis_tables` and `nis_names` man pages for additional information.

`_db_add`: child process attempting to add/modify

`_db_addib`: non-parent process attempting an add

These messages indicate that a read-only or nonparent process attempted to add or modify an object in the database. In most cases, these messages do not require any action on your part. If these messages are repeated frequently, call the Sun Solutions Center.

`db_checkpoint: Unable to checkpoint string-variable`

This message indicates that for some reason NIS+ was unable to complete checkpointing of a directory. The most likely cause is that the disk is full (See “Insufficient Disk Space” on page 338 for additional information).

`_db_remib: non-parent process attempting an remove`
`_db_remove: non-parent process attempting a remove`

These messages indicate that a read-only or non-parent process attempted to remove a table entry. In most cases, these messages do not require any action on your part. If these messages are repeated frequently, call the Sun Solutions Center.

`Do you want to see more information on this command?`

This indicates that there is a syntax or spelling error on your script command line.

`Entry/Table type mismatch`

This occurs when an attempt is made to add or modify an entry in a table, and the entry passed is of a different type from the table. For example, if the number of columns is not the same. Check that your update correctly matches the table type.

This message is generated by the NIS+ error code constant: `NIS_TYPEREMISMATCH`. See the `nis_tables` man page for additional information.

`error`

FNS error message. An error that cannot be classified as one of the other errors listed above occurred while processing the request. Check the status of the naming services involved in the operation and see whether any of them are experiencing extraordinary problems.

`**ERROR: chkey failed again. Please contact your network administrator to verify your network password.`

This message indicates that you typed the wrong network password.

- If this is the first time you are initializing this machine, contact your network administrator to verify the network password.

- If this machine has been initialized before as an NIS+ client of the same domain, try typing the root login password at the Secure RPC password prompt.
- If this machine is currently an NIS+ client and you are trying to change it to a client of a different domain, remove the `/etc/.rootkey` file, and then rerun the `nisclient` script, using the network password given to you by your network administrator (or the network password generated by the `nispopulate` script).

Error: Could not create a valid NIS+ coldstart file

This message is from `nisinit`, the NIS+ initialization routine. It is followed by another message preceded by a string that begins: "lookup:..". This second message will explain why a valid NIS+ cold-start file could not be created.

**ERROR: could not restore file *filename*

This message indicates that NIS+ was unable to rename *filename.no_nisplus* to *filename*.

Check your system console for system error messages.

- If there is a system error message, fix the problem described in the error message and then rerun `nisclient -i`.
- If there aren't any system error messages, try renaming this file manually, and then rerun `nisclient -i`.

**ERROR: Couldn't get the server *NIS+_server*'s address.

The script was unable to retrieve the server's IP address for the specified domain. Manually add the IP address for *NIS+_server* into the `/etc/hosts` file, then rerun `nisclient -i`.

**ERROR: directory *directory-path* does not exist.

This message indicates that you typed an incorrect directory path. Type the correct directory path.

**ERROR: *domainname* does not exist.

This message indicates that you are trying to replicate a domain that does not exist.

- If *domainname* is spelled incorrectly, rerun the script with the correct domain name.
- If the *domainname* domain does not exist, create it. Then you can replicate it.

**ERROR: *parent-domain* does not exist.

This message indicates that the parent domain of the domain you typed on the command line does not exist. This message should only appear when you are setting up a nonroot master server.

- If the domain name is spelled incorrectly, rerun the script with the correct domain name.
- If the domain's parent domain does not exist, you have to create the parent domain first, and then you can create this domain.

**ERROR: Don't know about the domain "*domainname*".
Please check your domainname.

This message indicates that you typed an unrecognized domain name.
Rerun the script with the correct domain name.

**ERROR: failed dumping *tablename* table.

The script was unable to populate the cred table because the script did not succeed in dumping the named table.

- If `niscat tablename.org_dir` fails, make sure that all the servers are operating, then rerun the script to populate the *tablename* table.
- If `niscat tablename.org_dir` is working, the error may have been caused by the NIS+ server being temporarily busy. Rerun the script to populate the *tablename* table.

****ERROR:** host *hostname* is not a valid NIS+ principal in domain *domainname*. This host *hostname* must be defined in the credential table in domain *domainname*. Use `nisclient -c` to create the host credential

A machine has to be a valid NIS+ client with proper credentials before it can become an NIS+ server. To convert a machine to an NIS+ root replica server, the machine first must be an NIS+ client in the root domain. Follow the instructions on how to add a new client to a domain, then rerun `nissserver -R`.

Before you can convert a machine to an NIS+ nonroot master or a replica server, the machine must be an NIS+ client in the parent domain of the domain that it plans to serve. Follow the instructions on how to add a new client to a domain, then rerun `nissserver -M` or `nissserver -R`.

This problem should not occur when you are setting up a root master server.

Error in accessing NIS+ cold start file is NIS+ installed?

This message is returned if NIS+ is not installed on a machine or if for some reason the file `/var/nis/NIS_COLD_START` could not be found or accessed. Check to see if there is a `/var/nis/NIS_COLD_START` file. If the file exists, make sure your path is set correctly and that `NIS_COLD_START` has the proper permissions. Then rename or remove the old cold-start file and rerun the `nisclient` script to install NIS+ on the machine.

This message is generated by the cache manager that sends the NIS+ error code constant: `NIS_COLDSTART_ERR`. See the `write` and `open` man pages for additional information on why a file might not be accessible.

Error in RPC subsystem

This fatal error indicates the RPC subsystem failed in some way. Generally, there will be a `syslog` message on either the client or server side indicating why the RPC request failed.

This message is generated by the NIS+ error code constant: `NIS_RPCERROR`. See the `nis_tables` and `nis_names` man pages for additional information.

****ERROR:** it failed to add the credential for root.

The NIS+ command `nisaddcred` failed to create the root credential when trying to set up a root master server. Check your system console for system error messages:

- If there is a system error message, fix the problem described in the error message and then rerun `nissserver`.
- If there aren't any system error messages, check to see whether the `rpc.nisd` process is running. If it is not running, restart it and then rerun `nissserver`.

****ERROR:** it failed to create the tables.

The NIS+ command `nissetup` failed to create the directories and tables. Check your system console for system error messages:

- If there is a system error message, fix the problem described in the error message and rerun `nissserver`.
- If there aren't any system error messages, check to see whether the `rpc.nisd` process is running. If it is not running, restart it and rerun `nissserver`.

****ERROR:** it failed to initialize the root server.

The NIS+ command `nisinit -r` failed to initialize the root master server. Check your system console for system error messages. If there is a system error message, fix the problem described in the error message and rerun `nissserver`.

****ERROR:** it failed to make the *domainname* directory

The NIS+ command `nismkdir` failed to make the new directory *domainname* when running `nissserver` to create a nonroot master. The parent domain does not have create permission to create this new domain.

- If you are not the owner of the domain or a group member of the parent domain, rerun the script as the owner or as a group member of the parent domain.
- If `rpc.nisd` is not running on the new master server of the domain that you are trying to create, restart `rpc.nisd`.

****ERROR:** it failed to promote new master for the *domainname* directory

The NIS+ command `nismkdir` failed to promote the new master for the directory *domainname* when creating a nonroot master with the `nisserver` script.

- If you do not have modify permission in the parent domain of this domain, rerun the script as the owner or as a group member of the parent domain.
- If `rpc.nisd` is not running on the servers of the domain that you are trying to promote, restart `rpc.nisd` on these servers and rerun `nisserver`.

****ERROR:** it failed to replicate the *directory-name* directory

The NIS+ command `nismkdir` failed to create the new replica for the directory *directory-name*.

- If `rpc.nisd` is not running on the master server of the domain that you are trying to replicate, restart `rpc.nisd` on the master server, rerun `nisserver`.
- If `rpc.nisd` is not running on the new replica server, restart it on the new replica and rerun `nisserver`.

****ERROR:** invalid group name.

It must be a group in the *root-domain* domain.

This message indicates that you used an invalid group name while trying to configure a root master server. Rerun `nisserver -r` with a valid group name for *root-domain*.

****ERROR:** invalid name "*client-name*"

It is neither an host nor an user name.

This message indicates that you typed an invalid *client-name*.

- If *client-name* was spelled incorrectly, rerun `nisclient -c` with the correct *client-name*.
- If *client-name* was spelled correctly, but it does not exist in the proper table, put *client-name* into the proper table and rerun `nisclient -c`. For example, a user client belongs in the `passwd` table, and a host client belongs in the `hosts` table.

****ERROR:** *hostname* is a master server for this domain. You cannot demote a master server to replica. If you really want to demote this master, you should promote a replica server to master using `nisserver` with the `-M` option.

You cannot directly convert a master server to a replica server of the same domain. You can, however, change a replica to be the new master server of a domain by running `nisserver -M` with the replica host name as the new master. This automatically makes the *old* master a replica.

****ERROR:** missing hostnames or usernames.

This messages indicates that you did not type the client names on the command line. Rerun `nisclient -c` with the client names.

****ERROR:** NIS+ group name must end with a "."

This message indicates that you did not specify a fully qualified group name ending with a period. Rerun the script with a fully qualified group name.

****ERROR:** NIS+ server is not running on *remote-host*. You must do the following before becoming a NIS+ server:

1. become a NIS+ client of the parent domain or any domain above the domain which you plan to serve. (`nisclient`)
2. start the NIS+ server. (`rpc.nisd`)

This message indicates that `rpc.nisd` is not running on the remote machine that you are trying to convert to an NIS+ server. Use the `nisclient` script to become an NIS+ client of the parent domain or any domain above the domain you plan to serve; start `rpc.nisd` on *remote-host*.

****ERROR:** `nisinit` failed.

`nisinit` was unable to create the `NIS_COLD_START` file.

Check the following:

- That the NIS+ server that you specified with the `-H` option is running—use `ping`
- That you typed the correct domain name
- That `rpc.nisd` is running on the server
- That the nobody class has read permission for this domain

****ERROR:** NIS map transfer failed.
tablename table will not be loaded.

NIS+ was unable to transfer the NIS map for this table to the NIS+ database.

- If the NIS server host is running, try running the script again. The error may have been due to a temporary failure.
- If all tables have this problem, try running the script again using a different NIS server.

****ERROR:** no permission to create directory *domainname*

The parent domain does not have create permission to create this new domain. If you are not the owner of the domain or as a group member of the parent domain, rerun the script as the owner, or as a group member of the parent domain.

****ERROR:** no permission to replicate directory *domainname*.

This message indicates that you do not have permission to replicate the domain. Rerun the script as the owner or as a group member of the domain.

****ERROR:** table *tablename.org_dir.domain* does not exist."
tablename table will not be loaded."

The script did not find the NIS+ table *tablename*.

- If *tablename* is spelled incorrectly, rerun the script with the correct table name.
- If the *tablename* table does not exist, use `nissetup` to create the table if *tablename* is one of the standard NIS+ tables. Or use `nistbladm` to create the private table *tablename*. Then rerun the script to populate this table.
- If the *tablename* table exists, the error may have been caused by the NIS+ server being temporarily busy. Rerun the script to populate this *tablename* table.

****ERROR:** this name "*client-name*" is in both the `passwd` and `hosts` tables.

You cannot have a username same as the hostname.

client-name appears in both the `passwd` and `hosts` tables. One name is not allowed to be in both of these tables. Manually remove the entry from either the `passwd` or `hosts` table. Then, rerun `nisclient -c`.

****ERROR:** You cannot use the `-u` option as a root user.

This message indicates that the superuser tried to run `nisclient -u`. The `-u` option is for initializing ordinary users only. Superusers do not need be initialized as NIS+ clients.

****ERROR:** You have specified the `Z` option after having selected the `X` option. Please select only one of these options [*list*]. Do you want to see more information on this command?

The script you are running allows you to choose only one of the listed options.

- Type `y` to view additional information.
- Type `n` to stop the script and exit.

After exiting the script, rerun it with just one of the options.

****ERROR:** you must specify a fully qualified groupname.

This message indicates that you did not specify a fully qualified group name ending with a period. Rerun the script with a fully qualified group name.

****ERROR:** you must specify both the NIS domainname (`-y`) and the NIS server hostname (`-h`).

This message indicates that you did not type either the NIS domain name and/or the NIS server host name. Type the NIS domain name and the NIS server host name at the prompt or on the command line.

****ERROR:** you must specify one of these options: `-c`, `-i`, `-u`, `-r`.

This message indicates that one of these options, `-c`, `-i`, `-u`, `-r` was missing from the command line. Rerun the script with the correct option.

****ERROR:** you must specify one of these options: `-r`, `-M` or `-R`

This message indicates that you did not type any of the `-r` or the `-M` or the `-R` options. Rerun the script with the correct option.

****ERROR:** you must specify one of these options: -C, -F, or -Y

This message indicates that you did not type either the -Y or the -F option. Rerun the script with the correct option.

****ERROR:** You must be root to use -i option.

This message indicates that an ordinary user tried to run `nisclient -i`. Only the superuser has permission to run `nisclient -i`.

Error while talking to callback proc

An RPC error occurred on the server while it was calling back to the client. The transaction was aborted at that time and any unsent data was discarded. Check the `syslog` on the server for more information.

This message is generated by the NIS+ error code constant: `NIS_CBERROR`. See the `nis_tables` man page for additional information.

First/Next chain broken

This message indicates that the connection between the client and server broke while a callback routine was posting results. This could happen if the server died in the middle of the process.

This message is generated by the NIS+ error code constant: `NIS_CHAINBROKEN`.

Generic system error

Some form of generic system error occurred while attempting the request. Check the `syslog` record on your system for error messages from the server.

This message usually indicates that the server has crashed or the database has become corrupted. This message may also be generated if you incorrectly specify the name of a server or replica as if it belonged to the domain it was servicing rather than the domain above. See “Domain Name Confusion” on page 310 for additional information.

This message is generated by the NIS+ error code constant: `NIS_SYSTEMERROR`. See the `nis_tables` and `nis_names` man pages for additional information.

illegal name

FNS error message. The name supplied is not a legal name.

Illegal object type for operation

See “Illegal Object Problems” on page 308 for a description of these type of problems.

This message is generated by the NIS+ error code constant: DB_BADOBJECT.

incompatible code sets

FNS error message. The operation involved character strings from incompatible code sets, or the supplied code set is not supported by the implementation.

insufficient permission to update credentials.

This message is generated by the `nisaddcred` command when you have insufficient permission to execute an operation. This could be insufficient permission at the table, column, or entry level. Use `niscat -o cred.org_dir` to determine what permissions you have for that cred table. If you need additional permission, you or the system administrator can change the permission requirements of the object as described in Chapter 7, “Administering NIS+ Access Rights,” or add you to a group that does have the required permissions as described in Chapter 9, “Administering NIS+ Groups.”

See “Ownership and Permission Problems” on page 318 for additional information about permission problems.

insufficient resources

FNS error message. The name service used by FNS does not have sufficient resources to complete the request. Check memory and disk availability on the name servers involved.

invalid attribute identifier

FNS error message. The attribute identifier is in a format not acceptable to the naming system, or its contents are not valid for the format specified for the identifier.

`invalid attribute value`

FNS error message. The value supplied is not in the correct form for the given attribute.

`invalid enumeration handle`

FNS error message. The enumeration handle supplied is invalid. The handle could have been from another enumeration, an update operation may have occurred during the enumeration, or there may have been some other reason.

`Invalid Object for operation`

- *Name context.* The name passed to the function is not a legal NIS+ name.
- *Table context.* The object pointed to is not a valid NIS+ entry object for the given table. This could occur if it had a mismatched number of columns, or a different data type (for example, binary or text) than the associated column in the table.

This message is generated by the NIS+ error code constant: `NIS_INVALIDOBJ`. See the `nis_tables` and `nis_names` man pages for additional information.

`invalid syntax attributes`

FNS error message. The syntax attributes supplied are invalid or insufficient to fully specify the syntax.

`invalid usecs`

Routine_name: `invalid usecs`

This message is generated when the value in the `tv_usec` field of a variable of type `struct time stamp` is larger than the number of microseconds in a second. This is usually due to some type of software error.

tablename is not a table

The object with the name *tablename* is not a table object. For example, the `nisgrep` and `nismatch` commands will return this error if the object you specify on the command line is not a table.

`link error`

FNS error message. An error occurred while resolving an XFN link with the supplied name.

`link loop limit reached`

FNS error message. A nonterminating loop was detected due to XFN links encountered during composite name resolution, or the implementation-defined limit was exceeded on the number of XFN links allowed for a single operation.

`Link Points to illegal name`

The passed name resolved to a LINK type object and the contents of the object pointed to an invalid name.

This message is generated by the NIS+ error code constant:

`NIS_LINKNAMEERROR`. See the `nis_tables` and `nis_names` man pages for additional information.

`Load limit of numeric-variable reached!`

An attempt has been made to create a child process when the maximum number of child processes have already been created on this server. This message is seen on the server's system log, but only if the threshold for logging messages has been set to include `LOG_WARNING` level messages.

`login and keylogin passwords differ.`

This message is displayed when you are changing your password with `nispaswd` and the system has changed your password, but has been unable to update your credential entry in the `cred` table with the new password and also unable to restore your original password in the `passwd` table. This message is followed by the instructions:

```
Use NEW password for login and OLD password for keylogin. Use
"chkey -p" to reencrypt the credentials with the new login
password. You must keylogin explicitly after your next login.
```

These instructions are then followed by a status message explaining why it was not possible to revert back to the old password. If you see these messages, be sure to follow the instructions as given.

Login incorrect

The most common cause of a “login incorrect” message is mistyping the password. Try it again. Make sure you know the correct password. Remember that passwords are case-sensitive (uppercase letters are considered different than lowercase letters) and that the letter “o” is not interchangeable with the numeral “0,” nor is the letter “1” the same as the numeral “1.”

For other possible causes of this message, see ““Login Incorrect” Message” on page 321.

log_resync: Cannot truncate transaction log file

An attempt has been made to checkpoint the log, and the `rpc.nisd` daemon is trying to shrink the log file after deleting the checkpointed entries from the log. See the `ftruncate` man pages for a description of various factors that might cause this routine to fail. See also “Namespace Database Problems” on page 311.

malformed link

FNS error message. A malformed link reference was found during a `fn_ctx_lookup_link()` operation. The name supplied resolved to a reference that was not a link.

Malformed Name or illegal name

The name passed to the function is not a legal or valid NIS+ name.

One possible cause for this message that someone changed an existing domain name. Existing domain names should not be changed. See “Changed Domain Name” on page 329.

This message is generated by the NIS+ error code constant: `NIS_BADNAME`. See the `nis_tables` man page for additional information.

_map_addr: RPC timed out.

A process or application could not contact NIS+ within its default time limit to get necessary data or resolve host names from NIS+. In most cases, this problem will solve itself after a short wait. See “Slow Performance and System Hang Problems” on page 332 for additional information about slow performance problems.

Master server busy full dump rescheduled

This message indicates that a replica server has been unable to update itself with a full dump from the master server because the master is busy. See “Replica Update Failure” on page 341 for additional information.

String Missing or malformed attribute

The name of an attribute did not match with a named column in the table, or the attribute did not have an associated value.

This could indicate an error in the syntax of a command. The *string* should give an indication of what is wrong. Common causes are spelling errors, failure to correctly place the equals sign (=), an incorrect column or table name, and so forth.

This message is generated by the NIS+ error code constant: NIS_BADATTRIBUTE. See the `nis_tables` man page for additional information.

Modification failed

Returned by the `nisgrpadm` command when someone else modified the group during the execution of your command. Check to see who else is working with this group. Reissue the command.

This message is generated by the NIS+ error code constant: NIS_IBMODERROR.

Modify operation failed

The attempted modification failed for some reason.

This message is generated by the NIS+ error code constant: NIS_MODFAIL. See the `nis_tables` and `nis_names` man pages for additional information.

name in use

FNS error message. The name supplied is already bound in the context.

name not found

FNS error message. The name supplied was not found.

Name not served by this server

A request was made to a server that does not serve the specified name. Normally this will not occur; however, if you are not using the built-in location mechanism for servers, you may see this if your mechanism is broken.

Other possible causes are:

- Cold-start file corruption. Delete the `/var/nis/NIS_COLD_START` file and then reboot.
- Cache problem such as the local cache being out of date. Kill the `nis_cachemgr` and `/var/nis/NIS_SHARED_DIRCACHE`, and then reboot. (If the problem is not in the root directory, you may be able to simply kill the domain cache manager and try the command again.)
- Someone removed the directory from a replica.

This message is generated by the NIS+ error code constant: `NIS_NOT_ME`. See the `nis_tables` and `nis_names` man pages for additional information.

Named object is not searchable

The table name resolved to an NIS+ object that was not searchable.

This message is generated by the NIS+ error code constant: `NIS_NOTSEARCHABLE`. See the `nis_tables` man page for additional information.

Name/entry isn't unique

An operation has been requested based on a specific search criteria that returns more than one entry. For example, you use `nistbladm -r` to delete a user from the `passwd` table, and there are two entries in that table for that user name as shown as follows:

```
mymachine# nistbladm -r [name=arnold],passwd.org_dir
Can't remove entry: Name/entry isn't unique
```

You can apply your command to multiple entries by using the `-R` option rather than `-r`. For example, to remove all entries for arnold:

```
mymachine# nistbladm -R name=arnold],passwd.org_dir
```

NIS+ error

The NIS+ server has returned an error, but the `passwd` command determines exactly what the error is.

NisDirCacheEntry:write: xdr_directory_obj failed

The most likely causes for this message is that an attempt to allocate system memory failed. See “Insufficient Memory” on page 337 for a discussion of memory problems. If your system does not seem to be short of memory, contact the Sun Solutions Center.

NIS+ operation failed

This generic error message should be rarely seen. Usually it indicates a minor software problem that the system can correct on its own. If it appears frequently, or appears to be indicating a problem that the system is not successfully dealing with, contact the Sun Solutions Center.

This message is generated by the NIS+ error code constant: `NIS_FAIL`.

String-variable: NIS+ server busy try again later.

See “Slow Performance and System Hang Problems” on page 332 for possible causes.

NIS+ server busy try again later.

Self explanatory. Try the command later.

See also “Slow Performance and System Hang Problems” on page 332 for possible causes.

NIS+ server for *string-variable* not responding still trying
See “Slow Performance and System Hang Problems” on page 332 for possible causes.

NIS+ server not responding
See “Slow Performance and System Hang Problems” on page 332 for possible causes.

NIS+ server needs to be checkpointed. Use `nisping -C domainname`

Checkpoint *immediately!* Do not wait!

This message is generated at the LOG_CRIT level on the server’s system log. It indicates that the log is becoming too large. Use `nisping -C domainname` to truncate the log by checkpointing.

See also “Logs Grow too Large” on page 309 for additional information on log size.

NIS+ servers unreachable

This soft error indicates that a server for the desired directory of the named table object could not be reached. This can occur when there is a network failure or the server has crashed. A new attempt may succeed. See the description of the HARD_LOOKUP flag in the `nis_tables` and `nis_names` man pages.

This message is generated by the NIS+ error code constant: NIS_NAMEUNREACHABLE.

NIS+ service is unavailable or not installed

Self-explanatory.

This message is generated by the NIS+ error code constant: NIS_UNAVAIL.

NIS+: write ColdStart File: `xdr_directory_obj` failed

The most likely causes for this message are:

- Bad or incorrect parameters. Check the syntax and accuracy of whatever command you most recently entered.
- An attempt to allocate system memory failed. See “Insufficient Memory” on page 337 for a discussion of memory problems.

- If your command syntax is correct, and your system does not seem to be short of memory, contact the Sun Solutions Center.

`nis_checkpoint_svc: readonly child instructed to checkpoint ignored.`

This is simply a status message indicating that a read-only process attempted to perform an operation restricted to the parent process, and the attempt was aborted. No action need be taken.

`nis_dumplog_svc: readonly child called to dump log, ignored`

This is simply a status message indicating that a read-only process attempted to perform an operation restricted to the parent process, and the attempt was aborted. No action need be taken.

`nis_dump_svc: load limit reached.`

The maximum number of child processes permitted on your system has been reached.

`nis_dump_svc: one replica is already resyncing.`

Only one replica can resync from a master at a time. Try the command later.

See “Replica Update Failure” on page 341 for information on these three error messages.

`nis_dump_svc: Unable to fork a process.`

The fork system call has failed. See the `fork` man page for possible causes.

`nis_mkdir_svc: readonly child called to mkdir, ignored`

This is simply a status message indicating that a read-only process attempted to perform an operation restricted to the parent process, and the attempt was aborted. No action need be taken.

`nis_ping_svc: readonly child was pung ignored.`

This is simply a status message indicating that a read-only process attempted to perform an operation restricted to the parent process, and the attempt was aborted. No action need be taken.

`nis_rmdir_svc: readonly child called to rmdir, ignored`

This is simply a status message indicating that a read-only process attempted to perform an operation restricted to the parent process, and the attempt was aborted. No action need be taken.

`nisaddcred: no password entry for uid userid`
`nisaddcred: unable to create credential.`

These two messages are generated during execution of the `nispopulate` script. The NIS+ command `nisaddcred` failed to add a LOCAL credential for the user ID *userid* on a remote domain. (This only happens when you are trying to populate the `passwd` table in a remote domain.)

To correct the problem, add a table path in the local `passwd` table:

```
# nistbladm -u -p passwd.org_dir.remote-domain passwd.org_dir
```

The *remote-domain* must be the same domain that you specified with the `-d` option when you ran `nispopulate`. Rerun the script to populate the `passwd` table.

No file space on server

Self-explanatory.

This message is generated by the NIS+ error code constant:
`NIS_NOFILESPACE`.

No match

This is most likely an error message from the shell, caused by failure to escape the brackets when specifying an indexed name. For example, failing to set off a bracketed indexed name with quote marks would generate this message because the shell would fail to interpret the brackets as shown as follows:

```
# nistbladm -m shell=/bin/csh [name=miyoko],passwd.org_dir  
No match
```

The correct syntax is:

```
# nistbladm -m shell=/bin/csh `[name=miyoko],passwd.org_dir`
```

No memory

Your system does not have enough memory to perform the specified operation. See “System Resource Problems” on page 337 for additional information on memory problems.

Non NIS+ namespace encountered

The name could not be completely resolved. This usually indicates that the name passed to the function resolves to a namespace that is outside the NIS+ name tree. In other words, the name is contained in an unknown directory. When this occurs, this error is returned with an NIS+ object of type DIRECTORY.

This message is generated by the NIS+ error code constant: NIS_FOREIGNNS. See the `nis_tables` or `nis_names` man pages for additional information.

No password entry for uid *userid*

No password entry found for uid *userid*

Both of these two messages indicate that no entry for this user was found in the passwd table when trying to create or add a credential for that user. (Before you can create or add a credential, the user must be listed in the passwd table.)

- The most likely cause is misspelling the user’s *userid* on the command line. Check your command line for correct syntax and spelling.
- Check that you are either in the correct domain, or specifying the correct domain on the command line.

- If the command line is correct, check the passwd table to make sure the user is listed under the *userid* you are entering. This can be done with `nismatch`:

```
mymachine# nismatch uid=userid passwd.org_dir.
```

If the user is not listed in the passwd table, use `nistbladm` or `nisaddent` to add the user to the passwd table before creating the credential.

`no permission`

FNS error message. The operation failed because of access control problems. See ““No Permission” Messages (FNS)” on page 344. See also “No Permission” on page 319.

`No shadow password information`

This means that password aging cannot be enforced because the information used to control aging is missing.

`no such attribute`

FNS error message. The object did not have an attribute with the given identifier.

`no supported address`

FNS error message. No shared library could be found under the `/usr/lib/fn` directory for any of the address types found in the reference bound to the FNS name. Shared libraries for an address type are named according to this convention: `fn_ctx_address_type.so`. Typically there is a link from `fn_ctx_address_type.so` to `fn_ctx_address_type.so.1`.

For example, a reference with address type `onc_fn_nisplus` would have a shared library in the path name:

```
/usr/lib/fn/fn_ctx_onc_fn_nisplus.so.
```

`not a context`

FNS error message. The reference does not correspond to a valid context.

Not found

String Not found

Names context. The named object does not exist in the namespace.

Table context. No entries in the table matched the search criteria. If the search criteria was null (return all entries), then this result means that the table is empty and may safely be removed.

If the `FOLLOW_PATH` flag was set, this error indicates that none of the tables in the path contain entries that match the search criteria.

This message is generated by the NIS+ error code constant: `NIS_NOTFOUND`. See the `nis_tables` and `nis_names` man pages for additional information.

See also “Object Not Found Problems” on page 314 for general information on this type of problem.

Not Found no such name

This hard error indicates that the named directory of the table object does not exist. This could occur when the server that should be the parent of the server that serves the table, does not know about the directory in which the table resides.

This message is generated by the NIS+ error code constant: `NIS_NOSUCHNAME`. See the `nis_names` and `nis_names` man pages for additional information.

See also “Object Not Found Problems” on page 314 for general information on this type of problem.

Not master server for this domain

This message may mean that an attempt was made to directly update the database on a replica server.

This message may also mean that a change request was made to a server that serves the name, but it is not the master server. This can occur when a directory object changes and it specifies a new master server. Clients that have cached copies of that directory object in their

`/var/nis/NIS_SHARED_DIRCACHE` file should run `ps` to obtain the

process ID of the `nis_cachemgr`, kill the `nis_cachemgr` process, remove the `/var/nis/NIS_SHARED_DICACHE` file, and then restart `nis_cachemgr`.

This message is generated by the NIS+ error code constant: `NIS_NOTMASTER`. See the `nis_tables` and `nis_names` man pages for additional information.

Not owner

The operation you attempted can only be performed by the object's owner, and you are not the owner.

This message is generated by the NIS+ error code constant: `NIS_NOTOWNER`.

operation not supported

FNS error message. The operation is not supported by the context. For example, trying to destroy an organization is not supported.

Object with same name exists

An attempt was made to add a name that already exists. To add the name, first remove the existing name and then add the new name or modify the existing named object.

This message is generated by the NIS+ error code constant: `NIS_NAMEEXISTS`. See the `nis_tables` and `nis_names` man pages for additional information.

parse error: *string-variable* (key *variable*)

This message is displayed by the `nisaddent` command when it attempts to use database files from a `/etc` directory and there is an error in one of the file's entries. The first variable should describe the problem, and the variable after `key` should identify the particular entry at fault. If the problem is with the `/etc/passwd` file, you can use `/usr/sbin/pwck` to check it.

partial result returned

FNS error message. The operation returned a partial result.

Partial Success

This result is similar to `NIS_NOTFOUND`, except that it means the request succeeded but resolved to zero entries.

When this occurs, the server returns a copy of the table object instead of an entry so that the client may then process the path or implement some other local policy.

This message is generated by the NIS+ error code constant: `NIS_PARTIAL`. See the `nis_tables` man page for additional information.

Passed object is not the same object on server

An attempt to remove an object from the namespace was aborted because the object that would have been removed was not the same object that was passed in the request.

This message is generated by the NIS+ error code constant: `NIS_NOTSAMEOBJ`. See the `nis_tables` and `nis_names` man pages for additional information.

Password does not decrypt secret key for *name*

Possible causes:

- You may have incorrectly typed the password.
- There may be no entry for *name* in the cred table.
- NIS+ could not decrypt the key (possibly because the entry might be corrupt).
- The Secure RPC password does not match the login password.
- The `nsswitch.conf` file may be directing the query to a local password in an `/etc/passwd` file that is different from the NIS+ password recorded in the cred table. (Note that the actual encrypted passwords are stored locally in the `/etc/shadow` file.)

See “Security Problems” on page 321 for information on diagnosing and solving these type of problem.

Password has not aged enough

This message indicates that your password has not been in use long enough and that you cannot change it until it has been in use for *N* (a number of) days. See “Changing Your Password” on page 140 for further information.

Permission denied

Returned when you do not have the permissions required to perform the operation you attempted. See “Ownership and Permission Problems” on page 318 for additional information.

This message might be related to a login or password matter, or an NIS+ security problem. The most common cause of a `Permission denied` message is that the password of the user receiving it has been locked by an administrator or the user’s account has been terminated. See Chapter 8, “Administering Passwords” and the “Security Problems” section of the “Problems and Solutions” appendix.

Permissions on the password database may be too restrictive

You do not have authorization to read (or otherwise use) the contents of the `passwd` field in an NIS+ table. See Chapter 7, “Administering NIS+ Access Rights,” for information on NIS+ access rights.

Please notify your System Administrator

When displayed as a result of an attempt to update password information with the `passwd` command, this message indicates that the attempt failed for one of many reasons. For example, the service might not be available, a necessary server is down, there is a “permission denied” type problem, and so forth. See “Security Problems” on page 321 for a discussion of various types of security problems.

Please check your `/etc/nsswitch.conf` file

The `nsswitch.conf` file specifies a configuration that is not supported for `passwd` update. See “`nsswitch.conf` File Requirements” on page 144 for supported configurations.

Probable success

Name context. The request was successful; however, the object returned came from an object cache and not directly from the server. (If you do not wish to see objects from object caches, you must specify the flag `NO_CACHE` when you call the lookup function.)

Table context. Even though the request was successful, a table in the search path was not able to be searched, so the result may not be the same as the one you would have received if that table had been accessible.

This message is generated by the NIS+ error code constant: NIS_S_SUCCESS. See the `nis_tables` and `nis_names` man pages for additional information.

Probably not found

The named entry does not exist in the table; however, not all tables in the path could be searched, so the entry may exist in one of those tables.

This message is generated by the NIS+ error code constant: NIS_S_NOTFOUND. See the `nis_tables` man page for additional information.

Query illegal for named table

A problem was detected in the request structure passed to the client library.

This message is generated by the NIS+ error code constant: NIS_BADREQUEST. See the `nis_tables` man page for additional information.

`replica_update`: Child process attempting update, aborted

This is simply a status message indicating that a read-only process attempted an update and the attempt was aborted.

`replica_update`: error result was *string*

This message indicates a problem (identified by *string*) in carrying out a dump to a replica. See “Replica Update Failure” on page 341 for further information.

`replica_update`: error result was Master server busy, full dump rescheduled

`replica_update`: master server busy rescheduling the resync.

`replica_update`: master server is busy will try later.

`replica_update`: nis dump result Master server busy, full dump rescheduled

These messages all indicate that the server is busy and the dump will be done later.

replica_update: nis dump result nis_perror *error string*

This message indicates a problem (identified by the *error string*) in carrying out a dump to a replica. See “Replica Update Failure” on page 341 for further information.

replica_update: *number* updates *number* errors

A status message indicating a successful update.

replica_update: WARNING: last_update(*directoryname*) returned 0!

A NIS+ process could not find the last update time stamp in the transaction log for that directory. This will cause the system to perform a full resync of the problem directory.

Results Sent to callback proc

This is simply a status message. No action need be taken.

This message is generated by the NIS+ error code constant: NIS_CBRESULTS. See the `nis_tables` man page for additional information.

root_replica_update: update failed *string-variable*: could not fetch object from master.

This message indicates a problem in carrying out a dump to a replica. See “Replica Update Failure” on page 341 for further information.

Security exception on local system. UNABLE TO MAKE REQUEST.

This message may be displayed if a user has the same login ID as a machine name. See “User Login Same as Machine Name” on page 319 for additional information.

Server busy, try again

The server was too busy to handle your request.

- For the add, remove, and modify operations, this message is returned when either the master server for a directory is unavailable or it is in the process of checkpointing its database.
- This message can also be returned when the server is updating its internal state.

- In the case of `nis_list`, if the client specifies a callback and the server does not have enough resources to handle the callback.

Retry the command at a later time when the server is available.

This message is generated by the NIS+ error code constant: `NIS_TRYAGAIN`. See the `nis_tables` and `nis_names` man pages for additional information.

Server out of memory

In most cases this message indicates a fatal result. It means that the server ran out of heap space.

This message is generated by the NIS+ error code constant: `NIS_NOMEMORY`. See the `nis_tables` and `nis_names` man pages for additional information.

Sorry

This message is displayed when a user is denied permission to login or change a password, and for security reasons the system does not display the reason for that denial because such information could be used by an unauthorized person to gain illegitimate access to the system.

Sorry: less than *N* days since the last change

This message indicates that your password has not been in use long enough and that you cannot change it until it has been in use for *N* days. See “Changing Your Password” on page 140 for further information.

Success

(1) The request was successful. This message is generated by the NIS+ error code constant: `NIS_SUCCESS`. See the `nis_tables` man page for additional information.

(2) FNS error message. Operation succeeded.

`_svcauth_des: bad nickname`

The nickname received from the client is invalid or corrupted, possibly due to network congestion. The severity of this message depends on what level of security you are running. At a low security level, this message is informational only; at a higher level, you may have to try the command again later.

`_svcauth_des: corrupted window from principal-name`

The window that was sent does not match the one sent in the verifier.

The severity of this message depends on what level of security you are running. At a low security level, this message is primarily for your information; at a higher level you may have to try the command again at some later time or take corrective action as described below.

Possible causes:

- The server's key pair has been changed. The client used the server's old public key while the server has a new secret key cached with `keyserv`. Run `keylogin` on both client and server.
- The client's key pair has been changed and the client has not run `keylogin` on the client system, so system is still sending the client's old secret key to the server, which is now using the client's new public key. Naturally, the two do not match. Run `keylogin` again on both client and server.
- Network corruption of data. Try the command again. If that does not work, use the `snoop` command to investigate and correct any network problems. Then run `keylogin` again on both server and client.

`_svcauth_des: decryption failure for principal-name`

DES decryption for some authentication data failed. Possible causes:

- Corruption to a library function or argument.
- A problem with a DES encryption chip, if you are using one.

The severity of this message depends on what level of security you are running. At a low security level, this message is primarily for your information; at a higher level, you may have to call the Sun Solutions Center for assistance. If the problem appears to be related to a DES encryption chip, call the Sun Solutions Center.

`svcauth_des: encryption failure`

DES encryption for some authentication data failed. Possible causes:

- Corruption of a library function or argument.
- A problem with a DES encryption chip, if you are using one.

The severity of this message depends on what level of security you are running. At a low security level, this message is primarily for your information; at a higher level, you may have to call Sun Solutions Center for assistance.

`_svcauth_des: invalid timestamp received from principal-name`

The time stamp received from the client is corrupted, or the server is trying to decrypt it using the wrong key. Possible causes:

- Congested network. Retry the command.
- Server cached out the entry for this client. Check the network load.

`_svcauth_des: key_decryptsessionkey failed for principal-name`

The `keyserv` process failed to decrypt the session key with the given public key. Possible causes are:

- The `keyserv` process is dead or not responding. Use `ps -ef` to check if the `keyserv` process is running on the `keyserv` host. If it is not, then restart it and run `keylogin`.
- The server principal has not keylogged in. Run `keylogin` for the server principal.
- The server principal (host) does not have credentials. Run `nismatch hostname.domainname.cred.org_dir` on the client's home domain cred table. Create new credentials if necessary.
- `keyserv` may have been restarted, in which case certain long-running applications, such as `rpc.nisd`, `sendmail`, and `automountd`, also need to be restarted.
- DES encryption failure. Call the Sun Solutions Center.

`_svcauth_des: no public key for principal-name`

The server cannot get the client's public key. Possible causes are:

- The principal has no public key. Run `nismatch` on the `cred` table of the principal's home domain. If there is no DES credential in that table for the principal, use `nisaddcred` to create one, and then run `keylogin` for that principal.
- The name service specified by a `nsswitch.conf` file is not responding.

`_svcauth_des: replayed credential from principal-name`

The server has received a request and finds an entry in its cache for the same client name and conversation key with the time stamp of the incoming request *before* that of the one currently stored in the cache.

The severity of this message depends on what level of security you are running. At a low security level, this message is primarily for your information. At a higher level, you may have to take corrective action as described below.

Possible causes are:

- The client and server clocks are out of sync. Use `rdate` to resync the client clock to the server clock.
- The server is receiving requests in random order. This could occur if you are using multithreading applications. If your applications support TCP, then set `/etc/netconfig` (or your `NETPATH` environment variable) to `tcp`.

`_svcauth_des: timestamp is earlier than the one previously seen from principal-name`

The time stamp received from the client on a subsequent call is earlier than one seen previously from that client. The severity of this message depends on what level of security you are running. At a low security level, this message is primarily for your information; at a higher level, you may have some corrective action as described below.

Possible causes are:

- The client and server clocks are out of sync. Use `rdate` to resync the client clock to the server clock.
- The server cached out the entry for this client. The server maintains a cache of information regarding the current clients. This cache size equals 64 client handles.

`_svcauth_des: timestamp expired for principal-name`

The time stamp received from the client is not within the default 35-second window in which it must be received. The severity of this message depends on what level of security you are running. At a low security level, this message is primarily for your information; at a higher level, you may have to take corrective action as described below.

Possible causes are:

- The 35-second window is too small to account for slow servers or a slow network.
- The client and server clocks are so far out of sync that the window cannot allow for the difference. Use `rdate` to resynchronize the client clock to the server clock.
- The server has cached out the client entry. Retry the operation.

`syntax not supported`

FNS error message. The syntax type is not supported.

`Too Many Attributes`

The search criteria passed to the server had more attributes than the table had searchable columns.

This message is generated by the NIS+ error code constant: `NIS_TOOMANYATTRS`. See the `nis_tables` man page for additional information.

`too many attribute values`

FNS error message. The operation attempted to associate more values with an attribute than the naming system supports.

`Too many failures - try later`

`Too many tries; try again later`

These messages refer to logging in or changing your password. They indicate that you have had too many failed attempts (or taken too long) to either log in or change your password. See “The Login incorrect Message” on page 139 or “Password Change Failures” on page 142 for further information.

Unable to authenticate NIS+ client

This message is generated when a server attempts to execute the callback procedure of a client and gets a status of `RPC_AUTHERR` from the `RPC clnt_call`. This is usually caused by out-of-date authentication information. Out-of-date authentication information can occur when the system is using data from a cache that has not been updated, or when there has been a recent change in the authentication information that has not yet been propagated to this server. In most cases, this problem should correct itself in a short period of time.

If this problem does not self-correct, it may indicate one of the following problems:

- Corrupted `/var/nis/NIS_SHARED_DIRCACHE` file. Kill the cache manager, remove this file, and restart the cache manager.
- Corrupted `/var/nis/NIS_COLD_START` file. Remove the file and then run `nisinit` to recreate it.
- Corrupted `/etc/.rootkey` file. Run `keylogin -r`.

This message is generated by the NIS+ error code constant: `NIS_CLNTAUTH`.

Unable to authenticate NIS+ server

In most cases, this is a minor software error from which your system should quickly recover without difficulty. It is generated when the server gets a status of `RPC_AUTHERR` from the `RPC clnt_call`.

If this problem does not quickly clear itself, it may indicate a corrupted `/var/nis/NIS_COLD_START`, `/var/nis/NIS_SHARED_DIRCACHE`, or `/etc/.rootkey` file.

This message is generated by the NIS+ error code constant: `NIS_SRVAUTH`.

Unable to bind to master server for name '*string-variable*'

See “Object Not Found Problems” on page 314 for information on this type of problem. This particular message may be caused by adding a trailing dot to the server’s domain name in the `/etc/defaultdomain` file.

Unable to create callback.

The server was unable to contact the callback service on your machine. This results in no data being returned.

See the `nis_tables` man page for additional information.

Unable to create process on server

This error is generated if the NIS+ service routine receives a request for a procedure number which it does not support.

This message is generated by the NIS+ error code constant: `NIS_NOPROC`.

String-variable: Unable to decrypt secret key for *string-variable*.

Possible causes:

- You may have incorrectly typed the password.
- There may be no entry for *name* in the cred table.
- NIS+ could not decrypt the key because the entry might be corrupt.
- The `nsswitch.conf` file may be directing the query to a local password in an `/etc/passwd` file that is different than the NIS+ password recorded in the cred table.

See “Security Problems” on page 321 for information on diagnosing and solving these type of problem.

unavailable

FNS error message. The name service on which the operation depends is unavailable.

Unknown error

This is displayed when the NIS+ error handling routine receives an error of an unknown type.

Unknown object

The object returned is of an unknown type.

This message is generated by the NIS+ error code constant: `NIS_UNKNOWNOBJ`. See the `nis_names` and `nis_names` man pages for additional information.

update_directory: *number* objects still running.

This is a status message displayed on the server during the update of a directory during a replica update. You do not need to take any action.

User *principalname* needs Secure RPC credentials to login but has none.

The user has failed to perform a keylogin. This problem usually arises when the user has different passwords in `/etc/shadow` and a remote NIS+ `passwd` table.

Warning: couldn't reencrypt secret key for <principal-name>

The most likely cause of this problem is that your Secure RPC password is different from your login password (or you have one password on file in a local `/etc/shadow` file and a different one in a remote NIS+ table) and you have not yet done an explicit `keylogin`. See “NIS+ Password and Login Password in `/etc/passwd` File” on page 330 and “Secure RPC Password and Login Passwords Are Different” on page 330 for more information on these types of problems.

WARNING: db::checkpoint: could not dump database: No such file or directory

This message indicates that the system was unable to open a database file during a checkpoint. Possible causes:

- The database file was deleted.
- The server is out of file descriptors.
- There is a disk problem
- You or the host do not have correct permissions.

WARNING: db_dictionary::add_table: could not initialize database from scheme

The database table could not be initialized. Possible causes:

- There was a system resource problem See “System Resource Problems” on page 337).
- You incorrectly specified the new table in the command syntax.
- The database is corrupted.

WARNING: db_query::db_query:bad index

In most cases this message indicates incorrect specification of an indexed name. Make sure that the indexed name is found in the specified table. Check the command for spelling and syntax errors.

**WARNING: domain *domainname* already exists.

This message indicates that the domain you tried to create already exists.

- If you are trying to promote a new nonroot master server or are recovering from a previous `nisserver` problem, continue running the script.
- If *domainname* was spelled incorrectly, rerun the script with the correct domain name.

**WARNING: failed to add new member *NIS+_principle* into the *groupname* group.

You will need to add this member manually:

```
1. /usr/sbin/nisgrpadm -a groupname NIS+_principal
```

The NIS+ command `nisgrpadm` failed to add a new member into the NIS+ group *groupname*. Manually add this NIS+ principal by typing:

```
# /usr/sbin/nisgrpadm -a groupname NIS+_principal
```

**WARNING: failed to populate *tablename* table.

The `nisaddent` command was unable to load the NIS+ *tablename* table. A more detailed error message usually appears before this warning message.

**WARNING: hostname specified will not be used.

It will use the local hostname instead.

This message indicates that you typed a remote host name with the `-H` option. The `nisserver -r` script does not configure remote machines as root master servers.

- If the local machine is the one that you want to convert to an NIS+ root master server, no other action is needed. The `nisserver -r` script will ignore the host name you typed.

- If you actually want to convert the remote host (instead of the local machine) to an NIS+ root master server, exit the script. Rerun the `nissserver -r` script on the remote host.

****WARNING:** *hostname* is already a server for this domain. If you choose to continue with the script, it will try to replicate the `groups_dir` and `org_dir` directories for this domain.

This is a message warning you that *hostname* is already a replica server for the domain that you are trying to replicate.

- If you are running the script to fix an earlier `nissserver` problem, continue running the script.
- If *hostname* was mistakenly entered, rerun the script with the correct host name.

****WARNING:** *alias-hostname* is an alias name for host *canonical_hostname*. You cannot create credential for host alias.

This message indicates that you have typed a host alias in the name list for `nisclient -c`. The script asks you if you want to create the credential for the canonical host name, since you should not create credentials for host alias names.

****WARNING:** file *directory-path/tablename* does not exist!
tablename table will not be loaded.

The script was unable to find the input file for *tablename*.

- If *directory-path/tablename* is spelled incorrectly, rerun the script with the correct table name.
- If the *directory-path/tablename* file does not exist, create and update this file with the proper data. Then rerun the script to populate this table.

****WARNING:** NIS `auto.master` map conversion failed.
`auto.master` table will not be loaded.

The `auto.master` map conversion failed while trying to convert all the dots to underscores in the `auto_master` table. Rerun the script with a different NIS server.

****WARNING:** NIS netgroup map conversion failed.
netgroup table will not be loaded.

The netgroup map conversion failed while trying to convert the NIS domain name to the NIS+ domain name in the netgroup map. Rerun the script with a different NIS server.

****WARNING:** nisupdkeys failed on directory *domainname*. This script will not be able to continue.
Please remove the *domainname* directory using 'nismkdir'.

The NIS+ command nisupdkeys failed to update the keys in the listed directory object. If rpc.nisd is not running on the new master server that is supposed to serve this new domain, restart rpc.nisd. Then use nismkdir to remove the *domainname* directory. Finally, rerun nisserver.

WARNING: nisupdkeys failed on directory *directory-name*
You will need to run nisupdkeys manually:

1. /usr/lib/nis/nisupdkeys *directory-name*

The NIS+ command nisupdkeys failed to update the keys in the listed directory object. Manually update the keys in the directory object by typing:

```
# /usr/lib/nis/nisupdkeys directory-name
```

****WARNING:** once this script is executed, you will not be able to restore the existing NIS+ server environment. However, you can restore your NIS+ client environment using "nisclient -r" with the proper domainname and server information. Use "nisclient -r" to restore your NIS+ client environment.

These messages appear if you have already run the script at least once before to set up an NIS+ server. They indicate that NIS+-related files will be removed and recreated as needed if you decide to continue running this script.

- If it is all right for these NIS+ files to be removed, continue running the script.
- If you want to save these NIS+ files, exit the script by typing *n* at the Do you want to continue? prompt. Then save the NIS+ files in a different directory and rerun the script.

****WARNING:** this script removes directories and files related to NIS+ under /var/nis directory with the exception of the NIS_COLD_START and NIS_SHARED_DIRCACHE files which will be renamed to <file>.no_nisplus. If you want to save these files, you should abort from this script now to save these files first.

See "WARNING: once this script is executed,..." above.

****WARNING:** you must specify the NIS domainname.

This message indicates that you did not type the NIS domain name at the prompt. Type the NIS server domain name at the prompt.

****WARNING:** you must specify the NIS server hostname. Please try again.

This message indicates that you did not type the NIS server host name at the prompt. Type the NIS server host name at the prompt.

Window verifier mismatch

This is a debugging message generated by the `_svcauth_des` code. A verifier could be invalid because a key was flushed out of the cache. When this occurs, `_svcauth_des` returns the `AUTH_BADCRED` status.

You (*string-variable*) do not have Secure RPC credentials in NIS+ domain '*string-variable*'

This message could be caused by trying to run `nispaswd` on a server that does not have the credentials required by the command. (Keep in mind that servers running at security level 0 do not create or maintain credentials.)

See "Ownership and Permission Problems" on page 318 for additional information on credential, ownership, and permission problems.

You may not change this password

This message indicates that your administrator has forbidden you to change your password.

You may not use nisplus repository

You used `-r nisplus` in the command line of your command, but the appropriate entry in the NIS+ passwd table was not found. Check the passwd table in question to make sure it has the entry you want. Try adding `nisplus` to the `nsswitch.conf` file.

Your password has been expired for too long

Your password is expired

These messages refer to password aging. They indicate that your password has been in use too long and needs to be changed now. See “The password expired Message” on page 139 for further information.

Your password will expire in *N* days

Your password will expire within 24 hours

These messages refer to password aging. They indicate that your password is about to become invalid and should be changed now. See “The will expire Message” on page 140 for further information.

Your specified repository is not defined in the `nsswitch` file!

This warning indicates that you have specified a password information repository with the `-r` option, but that password repository is not included in the `passwd` entry of the `nsswitch.conf` file. The command you have just used will perform its job and make whatever change you intend to the password information repository you specified with the `-r` flag. However, the change will be made to information that the `nsswitch.conf` file does not point to, so no one will ever gain the benefit of it until the switch file is altered to point to that repository.

For example, suppose the `passwd` entry of the switch file reads: `files nis`, and you used

```
passwd -r nisplus
```

to establish a password age limit. That limit would not affect anyone because they are still using a switch file set to `files nis`.

verify_table_exists: cannot create table for *string*
nis_perror *message*.

To perform an operation on a table, NIS+ first verifies that the table exists. If the table does not exist, NIS+ attempts to create it. If it cannot create the table, it returns this error message. The *string* portion of the message identifies the table that could not be located or created; the *nis_perror message* portion provides information as to the cause of the problem (you can look up that portion of the message as if it were an independent message in this appendix). Possible causes for this type of problem:

- The server was just added as a replica of the directory and it may not have the directory object. Run `nisping -C` to checkpoint.
- You are out of disk space. See “Insufficient Disk Space” on page 338.
- Database corruption.
- Some other type of software error. Contact the Sun Solutions Center.

Information in NIS+ Tables



This appendix summarizes the information stored in the following NIS+ tables:

<i>Auto_Home Table</i>	<i>page 404</i>
<i>Auto_Master Table</i>	<i>page 405</i>
<i>Bootparams Table</i>	<i>page 406</i>
<i>Cred Table</i>	<i>page 407</i>
<i>Ethers Table</i>	<i>page 408</i>
<i>Group Table</i>	<i>page 409</i>
<i>Hosts Table</i>	<i>page 410</i>
<i>Mail_aliases Table</i>	<i>page 411</i>
<i>Netgroup Table</i>	<i>page 411</i>
<i>Netmasks Table</i>	<i>page 413</i>
<i>Networks Table</i>	<i>page 414</i>
<i>Passwd Table</i>	<i>page 414</i>
<i>Protocols Table</i>	<i>page 416</i>
<i>RPC Table</i>	<i>page 416</i>
<i>Services Table</i>	<i>page 417</i>
<i>Timezone Table</i>	<i>page 418</i>

Without a name service, most network information would be stored in `/etc` files and almost all NIS+ tables have corresponding `/etc` files. With the NIS service, you stored network information in NIS maps that also mostly corresponded with `/etc` files.

In the Solaris environment the name service switch file (`nsswitch.conf`) allows you to specify one or more sources for different types of information. In addition to NIS+ tables, that source can be NIS maps, DNS zone files, or `/etc` tables. The order in which you specify them determines how the information from different sources is combined.

If you are creating input files for any of these tables, most tables share two formatting requirements:

- You must use one line per entry
- You must separate columns with one or more spaces or Tabs.

If a particular table has different or additional format requirements, they are described under the heading, “Input File Format. “

Auto_Home Table

The `auto_home` table is an indirect automounter map that enables an NIS+ client to mount the home directory of any user in the domain. It does this by specifying a mount point for each user’s home directory, the location of each home directory, and mount options, if any. Because it is an indirect map, the first part of the mount point is specified in the `auto_master` table, which is, by default, `/home`. The second part of the mount point (that is, the subdirectory under `/home`) is specified by the entries in the `auto_home` map, and is different for each user.

The `auto_home` table has two columns:

Table C-1 Auto_Home Table

Column	Content	Description
Key	Mount point	The login name of every user in the domain
Value	Options & location	The mount options for every user, if any, and the location of the user’s home directory

For example:

```
costas barcelona:/export/partition2/costas
```

The home directory of the user `costas`, which is located on the server `barcelona`, in the directory `/export/partition2/costas`, would be mounted under a client's `/home/costas` directory. No mount options were provided in the entry.

Auto_Master Table

The `auto_master` table lists all the automounter maps in a domain. For direct maps, the `auto_master` table provides a map name. For indirect maps, it provides both a map name and the top directory of its mount point. The `auto_master` table has two columns:

Table C-2 Auto_Master Table

Column	Content	Description
Key	Mount point	The top directory into which the map will be mounted. If the map is a direct map, this is a dummy directory, represented with <code>/-</code> .
Value	Map name	The name of the automounter map

For example, assume these entries in the `auto_master` table:

```
/home auto_home
/-auto_man
/programs auto_programs
```

The first entry names the `auto_home` map. It specifies the top directory of the mount point for all entries in the `auto_home` map: `/home`. (The `auto_home` map is an indirect map.) The second entry names the `auto_man` map. Because that map is a direct map, the entry provides only the map name. The `auto_man` map will itself provide the topmost directory, as well as the full pathname, of the mount points for each of its entries. The third entry names the `auto_programs` map and, since it provides the top directory of the mount point, the `auto_programs` map is an indirect map.

All automounter maps are stored as NIS+ tables. By default, the Solaris environment provides the `auto_master` map, which is mandatory, and the `auto_home` map, which is a great convenience. You can create more automounter maps for a domain, but be sure to store them as NIS+ tables and list them in the `auto_master` table. For more information about the automounter consult books about the automounter or books that describe the NFS file system.

Bootparams Table

The `bootparams` table stores configuration information about every diskless workstation in a domain. A diskless workstation is a workstation that is connected to a network, but has no hard disk. Since it has no internal storage capacity, a diskless workstation stores its files and programs in the file system of a server on the network. It also stores its configuration information—or *boot parameters*—on a server.

Because of this arrangement, every diskless workstation has an initialization program that knows where this information is stored. If the network has no name service, the program looks for this information in the server's `/etc/bootparams` file. If the network uses the NIS+ name service, the program looks for it in the `bootparams` table, instead.

The `bootparams` table can store any configuration information about diskless workstations. It has two columns: one for the configuration key, another for its value. By default, it is set up to store the location of each workstation's root, swap, and dump partitions.

The default `bootparams` table has only two columns that provide the following items of information:

Table C-3 Bootparams Table

Column	Content	Description
Key	Hostname	The diskless workstation's official host name, as specified in the hosts table
Value	Configuration	Root partition: the location (server name and path) of the workstation's root partition Swap partition: the location (server name and path) of the workstation's swap partition

Table C-3 Bootparams Table

Column	Content	Description
		<i>Dump partition:</i> the location (server name and path) of the workstation's dump partition
		<i>Install partition.</i>
		<i>Domain.</i>

Input File Format

The columns are separated with a TAB character. Backslashes (\) are used to break a line within an entry. The entries for root, swap, and dump partitions have the following format:

```
client-name root=server:path \  

  swap=server:path \  

  dump=server:path \  

  install=server:path \  

  domain=domainname
```

Here is an example:

```
buckaroo root=bigriver:/export/root1/buckaroo \  

  swap=bigriver:/export/swap1/buckaroo \  

  dump=bigriver:/export/dump/buckaroo \  

  install=bigriver:/export/install/buckaroo \  

  domain=sales.wiz.com
```

Additional parameters are available for x86-based workstations. See the `bootparams` man page for additional information.

Cred Table

The cred table stores credential information about NIS+ principals. Each domain has one cred table, which stores the credential information of client workstations that belong to that domain and client users who are allowed to log into them. (In other words, the principals of that domain.) The cred tables are located in their domains' `org_dir` subdirectory.

Note – Do not link a cred table. Each `org_dir` directory should have its own cred table. Do not use a link to some other `org_dir` cred table.

The cred table has five columns:

Table C-4 Cred Table

NIS+ Principal Name	Authentication Type	Authentication Name	Public Data	Private Data
Principal name of a principal user	LOCAL	UID	GID list	
Principal name of a principal user or workstation	DES	Secure RPC netname	Public key	Encrypted private key

The second column, authentication type, determines the types of values found in the other four columns.

- *LOCAL*. If the authentication type is LOCAL, the other columns contain a principal user’s name, UID, and GID; the last column is empty.
- *DES*. If the authentication type is DES, the other columns contain a principal’s name, Secure RPC netname, public key, and encrypted private key. These keys are used in conjunction with other information to encrypt and decrypt a DES credential.

See Chapter 5, “Administering NIS+ Credentials,” for additional information on credentials and the cred table.

Ethers Table

The ethers table stores information about the 48-bit Ethernet addresses of workstations on the Internet. It has three columns:

Table C-5 Ethers Table

Column	Content	Description
Addr	Ethernet-address	The 48-bit Ethernet address of the workstation
Name	Official-host-name	The name of the workstation, as specified in the hosts table
Comment	Comment	An optional comment about the entry

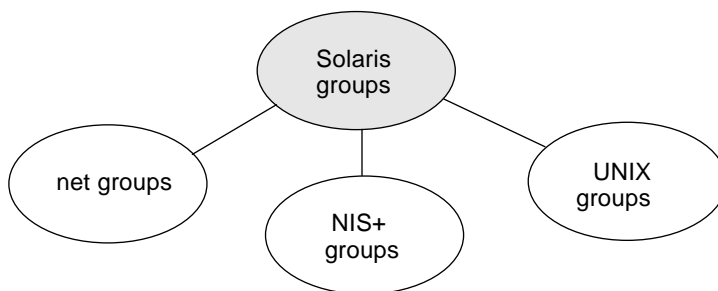
An Ethernet address has the form:

n:n:n:n:n:n hostname

where *n* is a hexadecimal number between 0 and FF, representing one byte. The address bytes are always in network order (most significant byte first).

Group Table

The group table stores information about workstation user groups. In the Solaris environment, three kinds of groups: net groups, NIS+ groups, and UNIX groups.



A net group is a group of workstations and users that have permission to perform remote operations on other workstations in the group. An NIS+ group is a set of NIS+ users that can be assigned access rights to an NIS+ object. They are described in Chapter 4, “Security Overview.” A UNIX group is simply a collection of users who are given additional UNIX access permissions.

UNIX groups allow a set of users on the network to access a set of files on several workstations or servers without making those files available to everyone. For example, the engineering and marketing staff working on a particular project could form a workstation user group.

The group table has four columns:

Table C-6 Group Table

Column	Description
Name	The group's name
Passwd	The group's password
GID	The group's numerical ID
Members	The names of the group members, separated by commas

Previous Solaris releases used a +/- syntax in local `/etc/group` files to incorporate or overwrite entries in the NIS group maps. Since the Solaris environment uses the name service switch file to specify a workstation's sources of information, this is no longer necessary. All you have to do in Solaris 2 systems is edit a client's `/etc/nsswitch.conf` file to specify files, followed by `nisplus` as the sources for the group information. This effectively adds the contents of the group table to the contents of the client's `/etc/group` file.

Hosts Table

The hosts table associates the names of all the workstations in a domain with their IP addresses. The workstations are usually also NIS+ clients, but they don't have to be. Other tables, such as `bootparams`, `group`, and `netgroup`, rely on the network names stored in this table. They use them to assign other attributes, such as home directories and group memberships, to individual workstations. The hosts table has four columns:

Table C-7 Hosts Table

Column	Description
Addr	The workstation's IP address (network number plus workstation ID number)
Cname	The workstation's official name
Name	A name used in place of the host name to identify the workstation
Comment	An optional comment about the entry

Mail_aliases Table

The `mail_aliases` table lists the domain's mail aliases recognized by `sendmail`. It has four columns:

Table C-8 Mail_aliases Table

Column	Description
Alias	The name of the alias
Expansion	A list containing the members that receive mail sent to this alias; members can be users, workstations, or other aliases
Comment	An optional comment about the entry
Options	(See man page for options)

Input File Format

Each entry has the following format:

```
alias-name: member[ , member] . . .
```

To extend an entry over several lines, use a backslash.

Netgroup Table

The `netgroup` table defines network wide groups used to check permissions for remote mounts, logins, and shells. The members of net groups used for remote mounts are workstations; for remote logins and shells, they are users.

Note – Users working on a client machine being served by a NIS+ server running in compatibility mode cannot run `ypcat` on the `netgroup` table. Doing so will give you results as if the table were empty even if it has entries.

The netgroup table has six columns:

Table C-9 Netgroup Table

Column	Content	Description
Name	groupname	The name of the network group
Group	groupname	Another group that is part of this group
Host	hostname	The name of a host
User	username	A user's login name
Domain	domainname	A domain name
Comment	Comment	An optional comment about the entry

Input File Format

The input file consists of a group name and any number of members:

```
groupname member-list...
```

The member list can contain the names of other net groups or an ordered member list with three fields or both:

```
member-list:=groupname | (hostname, username, domainname)
```

The first field of the member list specifies the name of a workstation that belongs to the group. The second field specifies the name of a user that belongs to the group. The third field specifies the domain in which the member specification is valid.

A missing field indicates a wildcard. For example, this net group includes all workstations and users in all domains:

```
everybody (,)
```

A dash in a field is the opposite of a wildcard; it indicates that no workstations or users belong to the group. Here are two examples:

```
(host1, -,wiz.com.)
(-,joe,wiz.com.)
```

The first specification includes one workstation, `host1`, in the `wiz.com.` domain, but excludes all users. The second specification includes one user in the `wiz.com.` domain, but excludes all workstations.

Netmasks Table

The netmasks table contains the network masks used to implement standard Internet subnetting. The table has three columns:

Table C-10 Netmasks Table

Column	Description
Addr	The IP number of the network
Mask	The network mask to use on the network
Comment	An optional comment about the entry

For network numbers, you can use the conventional IP dot notation used by workstation addresses, but leave zeroes in place of the workstation addresses. For example, this entry

```
128.32.0.0      255.255.255.0
```

means that class B network `128.32.0.0` should have 24 bits in its subnet field, and 8 bits in its host field.

Networks Table

The networks table lists the networks of the Internet. This table is normally created from the official network table maintained at the Network Information Control Center (NIC), though you may need to add your local networks to it. It has four columns:

Table C-11 Networks Table

Column	Description
Cname	The official name of the network, supplied by the Internet
Addr	The official IP number of the network
Name	An unofficial name for the network
Comment	An optional comment about the entry

Passwd Table

The passwd table contains information about the accounts of users in a domain. These users generally are, but do not have to be, NIS+ principals. Remember though, that if they are NIS+ principals, their credentials are not stored here, but in the domain's cred table. The passwd table usually grants read permission to the world (or to nobody).

Note – There should not be any entry in this table for the user root (user ID 0). Root's password information should be stored and maintained in the machine's `/etc` files.

The information in the passwd table is added when users' accounts are created.

The passwd table contains the following columns:

Table C-12 Passwd Table

Column	Description
Name	The user's login name, which is assigned when the user's account is created; the name can contain no uppercase characters and can have a maximum of eight characters
Passwd	The user's encrypted password
UID	The user's numerical ID, assigned when the user's account is created
GID	The numerical ID of the user's default group
GCOS	The user's real name plus information that the user wishes to include in the From: field of a mail-message heading; an "&" in this column simply uses the user's login name
Home	The path name of the user's home directory.
Shell	The user's initial shell program; the default is the Bourne shell: <code>/usr/bin/sh</code> .
Shadow	(See Table C-13 on page 415.)

The passwd table shadow column stores restricted information about user accounts. It includes the following information:

Table C-13 Passwd Table Shadow Column

Item	Description
Lastchg	The number of days between January 1, 1970, and the date the password was last modified
Min	The minimum number of days recommended between password changes
Max	The maximum number of days that the password is valid
Warn	The number of days' warning a user receives before being notified that his or her password has expired
Inactive	The number of days of inactivity allowed for the user
Expire	An absolute date past which the user's account is no longer valid
Flag	Reserved for future use: currently set to 0.

Previous Solaris releases used a +/- syntax in local `/etc/passwd` files to incorporate or overwrite entries in the NIS password maps. Since the Solaris 2 environment uses the name service switch file to specify a workstation's sources of information, this is no longer necessary. All you have to do in Solaris 2 systems is edit a client's `/etc/nsswitch.conf` file to specify files, followed by `nisplus` as the sources for the `passwd` information. This effectively adds the contents of the `passwd` table to the contents of the `/etc/passwd` file.

However, if you still want to use the +/- method, edit the client's `nsswitch.conf` file to add `compat` as the `passwd` source if you are using NIS. If you are using NIS+, add `passwd_compat: nisplus`.

Protocols Table

The protocols table lists the protocols used by the Internet. It has four columns:

Table C-14 Protocols Table

Column	Description
Cname	The protocol name
Name	An unofficial alias used to identify the protocol
Number	The number of the protocol
Comments	Comments about the protocol

RPC Table

The RPC table lists the names of RPC programs. It has four columns:

Table C-15 RPC Table

Column	Description
Cname	The name of the program
Name	Other names that can be used to invoke the program
Number	The program number
Comments	Comments about the RPC program

Here is an example of an input file for the RPC table:

```
#
# rpc file
#
rpcbind      100000      portmap      sunrpc      portmapper
rusersd      100002      rusers
nfs          100003      nfsprog
mountd       100005      mount        showmount
walld        100008      rwall        shutdown
sprayd       100012      spray
llockmgr     100020
nlockmgr     100021
status       100024
bootparam    100026
keyserv      100029      keyserver
nisd         100300      rpc.nisd
#
```

Services Table

The services table stores information about the Internet services available on the Internet. It has five columns:

Table C-16 Services Table

Column	Description
Cname	The official Internet name of the service
Name	The list of alternate names by which the service can be requested
Proto	The protocol through which the service is provided (for instance, 512/tcp)
Port	The port number
Comment	Comments about the service

Timezone Table

The timezone table lists the default timezone of every workstation in the domain. The default time zone is used during installation but can be overridden by the installer. The table has three columns:

Table C-17 Timezone Table

Field	Description
Name	The name of the domain
Tzone	The name of the time zone (for example, US/Pacific)
Comment	Comments about the time zone

Glossary

access rights

The permissions assigned to classes of NIS+ principals that determine what operations they can perform on NIS+ objects: read, modify, create or destroy.

authentication

The determination of whether an NIS+ server can identify the sender of a request for access to the NIS+ namespace. Authenticated requests are divided into the authorization categories of owner, group, and world. Unauthenticated requests—the sender is unidentified, are placed in the Nobody category. Whether or not such a request is granted depends upon the access rights given to a particular category.

authorization

The determination of the access rights of a particular category of authenticated user. The categories are owner, group, world and nobody. The possible rights a category could have are to read, modify, create and destroy an NIS+ object.

cache manager

The program that manages the local caches of NIS+ clients (`NIS_SHARED_DIRCACHE`), which are used to store location information about the NIS+ servers that support the directories most frequently used by those clients, including transport addresses, authentication information, and a time-to-live value.

child domain

See *domain*.

client

(1) In NIS+, the client is a principal (machine or user) requesting an NIS+ service from an NIS+ server.

(2) In the client-server model for file systems, the client is a machine that remotely accesses resources of a compute server, such as compute power and large memory capacity.

(3) In the client-server model, the client is an *application* that accesses services from a “server process.” In this model, the client and the server can run on the same machine or on separate machines.

client-server model

A common way to describe network services and the model user processes (programs) of those services. Examples include the name-server/name-resolver paradigm of the *Domain Name System (DNS)* and file-server/file-client relationships such as *NFS* and diskless hosts. See also *client*.

cold-start file

The NIS+ file given to a client when it is initialized that contains sufficient information so that the client can begin to contact the master server in its home domain.

credentials

The authentication information about an NIS+ principal that the client software sends along with each request to an NIS+ server. This information verifies the identity of a user or machine.

data encrypting key

A key used to encipher and decipher data intended for programs that perform encryption. Contrast with *key encrypting key*.

data encryption standard (DES)

A commonly used, highly sophisticated algorithm developed by the U.S. National Bureau of Standards for encrypting and decrypting data. See also *SUN-DES-1*.

decimal dotted notation

The syntactic representation for a 32-bit integer that consists of four 8-bit numbers written in base 10 with periods (dots) separating them. Used to represent IP addresses in the Internet as in: 192.67.67.20.

DES

See *data encryption standard (DES)*.

directory

(1) An NIS+ directory is a container for NIS+ objects such as NIS+ tables, groups, or subdirectories

(1) In UNIX a container for files and subdirectories.

directory cache

A local file used to store data associated with directory objects.

DNS

See Domain Name System.

DNS-forwarding

An NIS server or an NIS+ server with NIS compatibility set forwards requests it cannot answer to DNS servers.

DNS zones

Administrative boundaries within a network domain, often made up of one or more subdomains.

DNS zone files

A set of files wherein the DNS software stores the names and IP addresses of all the workstations in a domain.

domain

(1) In NIS+ a group of hierarchical objects managed by NIS+. There is one highest level domain (root domain) and zero or more subdomains. Domains and subdomains may be organized around geography, organizational or functional principles.

- *Parent domain.* Relative term for the domain immediately above the current domain in the hierarchy.
- *Child domain.* Relative term for the domain immediately below the current domain in the hierarchy.
- *Root domain.* Highest domain within the current NIS+ hierarchy.

(2) In the Internet, a part of a naming hierarchy. Syntactically, an Internet domain name consists of a sequence of names (labels) separated by periods (dots). For example, tundra.mpk.ca.us.

(3) In International Organization for Standardization's open systems interconnection (OSI), "domain" is generally used as an administrative partition of a complex distributed system, as in MHS private management domain (PRMD), and directory management domain (DMD).

domain name

The name assigned to a group of systems on a local network that share administrative files. The domain name is required for the network information service database to work properly. See also *domain*.

Domain Name System (DNS)

The network information service used by the Internet.

encryption key

See *data encrypting key*.

entry

A single row of data in a database table.

GID

See group ID.

group

(1) A collection of users who are referred to by a common name.

(2) In NIS+ a collection of users who are collectively given specified access rights to NIS+ objects. NIS+ group information is stored in the NIS+ group table.

(3) In UNIX, groups determine a user's access to files. There are two types of groups: default user group and standard user group.

group ID

A number that identifies the default *group* for a user.

indexed name

A naming format used to identify an entry in a table.

internet

A collection of networks interconnected by a set of routers that enable them to function as a single, large virtual network.

Internet

The world network of networks. The Internet uses the Internet protocol suite.

Internet address	A 32-bit address assigned to hosts using <i>TCP/IP</i> . See <i>decimal dotted notation</i> .
IP	Internet Protocol. The <i>network layer</i> protocol for the Internet protocol suite.
IP address	A unique number that identifies each host in a network.
key (column)	An NIS+ table entry's data can be accessed from any column, regardless of that table's key.
key, encrypting	A key used to encipher and decipher other keys, as part of a key management and distribution system. Contrast with <i>data encrypting key</i> .
key server	A Solaris process that stores private keys.
local-area network (LAN)	Multiple systems at a single geographical site connected together for the purpose of sharing and exchanging data and software.
mail exchange records	Files that contain a list of DNS domain names and their corresponding mail hosts.
mail hosts	A workstation that functions as an email router and receiver for a site.
master server	The server that maintains the master copy of the network information service database for a particular domain. Namespace changes are always made to the name service database kept by the domain's master server. Each domain has only <i>one</i> master server.
MIS	Management information systems (or services)
name resolution	The process of translating workstation or user names to addresses.

name server

Servers that run the DNS software and store the names and addresses of the workstations in the domain.

name service switch

A configuration file (`/etc/nsswitch.conf`) that defines the sources from which an NIS+ client can obtain its network information.

namespace

(1) A namespace stores information that users, workstations, and applications must have to communicate across the network.

(2) *NIS+ namespace*. A collection of hierarchical network information used by the NIS+ software.

(3) *NIS namespace*. A collection of *non*-hierarchical network information used by the NIS software.

(4) *DNS namespace*. A collection of networked workstations that use the DNS software.

network mask

A number used by software to separate the local subnet address from the rest of a given Internet protocol address.

network password

See Secure RPC password.

NIS

A distributed network information service containing key information about the systems and the users on the network. The NIS database is stored on the *master server* and all the *replica* or *slave servers*.

NIS maps

A file used by NIS that holds information of a particular type, for example, the password entries of all users on a network or the names of all host machines on a network. Programs that are part of the NIS service query these maps. See also *NIS*.

NIS+

A distributed network information service containing hierarchical information about the systems and the users on the network. The NIS+ database is stored on the *master server* and all the *replica servers*.

NIS-compatibility mode

A configuration of NIS+ that allows NIS clients to have access to the data stored in NIS+ tables. When in this mode, NIS+ servers can answer requests for information from both NIS and NIS+ clients.

NIS+ environment

For administrative purposes, an NIS+ environment refers to any situation in which the applicable `nsswitch.conf` file points to `nisplus`. Or any time a command is run with an option that forces it to operate on objects in an NIS+ namespace (for example, `passwd -r nisplus`).

NIS+ object

An NIS+ domain, directory, table, or group. See *domain*, *directory*, *group*, and *table*.

NIS+ principal

See *principal*.

NIS+ transaction log

A file that contains data updates destined for the NIS+ tables about objects in the namespace. Changes in the namespace are stored in the transaction log until they are propagated to replicas. The transaction log is cleared only after all of a master server's replicas have been updated.

parent domain

See *domain*.

principal

Any user of NIS+ information whose credentials have been stored in the namespace. Any user or machine that can generate a request to a NIS+ server. There are two kinds of NIS+ principal: client users and client machines:

- *Root principal*. A machine root user (user ID = 0). Requires only a DES credential.
- *User principal*. Any nonroot user (user ID > 0). Requires local and DES credentials.

private key

The private component of a pair of mathematically generated numbers, which, when combined with a private key, generates the DES key. The DES key in turn is used to encode and decode information. The private key of the sender is only available to the owner of the key. Every user or machine has its own public and private key pair.

public key

The public component of a pair of mathematically generated numbers, which, when combined with a private key, generates the DES key. The DES key in turn is used to encode and decode information. The public key is available to all users and machines. Every user or machine has their own public and private key pair.

populate tables

Entering data into NIS+ tables either from files or from NIS maps.

record

See *entry*.

remote procedure call (RPC)

An easy and popular paradigm for implementing the client-server model of distributed computing. A request is sent to a remote system to execute a designated procedure, using arguments supplied, and the result is returned to the caller.

replica server

NIS+ server that maintains a duplicate copy of the domain's master NIS+ server database. Replicas run NIS+ server software and maintain copies of NIS+ tables. A replica server increases the availability of NIS+ services. Each NIS+ domain should have at least one, and perhaps more, replicas. (In an NIS namespace, a replica server was known as a *slave* server.)

reverse resolution

The process of converting workstation IP addresses to workstation names using the DNS software.

root domain

See *domain*.

root master server

The master server for a NIS+ root domain.

root replica server

NIS+ server that maintains a duplicate copy of the root domain's master NIS+ server database.

RPC

See *remote procedure call (RPC)*.

server

(1) In NIS+ a host machine providing NIS+ services to NIS+ principals (machines and users).

(2) In the *client-server model* for file systems, the server is a machine with compute resources (and is sometimes called the compute server), and large memory capacity. Client machines can remotely access and make use of these resources. In the client-server model for window systems, the server is a process that provides windowing services to an application, or “client process.” In this model, the client and the server can run on the same machine or on separate machines.

(3) A *daemon* that actually handles the providing of files.

Secure RPC password

Password required by Secure RPC protocol. This password is used to encrypt the private key. This password should always be identical to the user’s login password.

slave server

(1) A server system that maintains a copy of the NIS database. It has a disk and a complete copy of the operating system.

(2) Slave servers are called *replica servers* in NIS+.

subnet

A working scheme that divides a single logical network into smaller physical networks to simplify routing.

table

In NIS+ a two-dimensional (nonrelational) database object containing NIS+ data in rows and columns. (In NIS an NIS map is analogous to a NIS+ table with two columns.) A table is the format in which NIS+ data is stored. NIS+ provides 16 predefined or system tables. Each table stores a different type of information.

TCP

See *Transport Control Protocol (TCP)*.

TCP/IP

Acronym for Transport Control Protocol/Interface Program. The protocol suite originally developed for the Internet. It is also called the *Internet* protocol suite. Solaris networks run on TCP/IP by default.

Transport Control Protocol (TCP)

The major transport protocol in the Internet suite of protocols providing reliable, connection-oriented, full-duplex streams. Uses IP for delivery. See *TCP/IP*.

wide-area network (WAN)

A network that connects multiple local-area networks (LANs) or systems at different geographical sites via phone, fiber-optic, or satellite links.

Index

Symbols

- # (hash character)
 - nsswitch.conf file comments, 238
 - shell prompts, xxxii
- \$ (dollar sign)
 - NIS_PATH variable use, 44
 - shell prompts, xxxii
- " (quotation marks) in fully qualified names, 42
- * (asterisk)
 - in group member listing, 176
 - regular expression symbol, 220
- (dash), in netgroup table input file format, 413
- (minus sign)
 - access rights operator, 121, 122
 - group names, 175
 - nsswitch.conf compatibility with +/- syntax, 235, 242 to 243
- + (plus sign)
 - access rights operator, 121, 122
 - nsswitch.conf compatibility with +/- syntax, 235, 242 to 243
 - regular expression symbol, 220
- . (dots)
 - ending domain names, 315
 - ending NIS+ object names, 24
 - in /etc/auto* table names, 318
 - host names and, 41 to 42
 - NIS+ namespace object names, 23
 - NIS+ principal names vs. Secure RPC netnames, 42
 - regular expression symbols, 220
- / (slashes)
 - double slashes in organization names, 283
 - trailing, in organization names, 284
 - in UNIX filenames, 23
- = (equals sign) as access rights operator, 121, 122
- ? (question mark) as regular expression symbol, 220
- @ (at sign) in group member listing, 176
- \ (backslash)
 - bootparams table input file format, 407
 - regular expression symbol, 220
- | (pipe symbol) as regular expression symbol, 220

Numerics

- 0 security level
 - described, 59
 - servers running at level 0, 319
- 1 security level, 59

2 security level, 59

A

-A

- nisaddent option, 227
- nischttl option, 204, 205
- nistbladm option, 211, 213

-a

- nisaddent option, 226, 227, 228, 230
- nisdefault option, 124, 125
- nisgrpadm option, 177, 179 to 180
- nisping option, 200
- nistbladm option, 211 to 212
- nisupdkeys option, 107
- rpc.nispasswd option, 171

a access rights, 121

“abort_transaction” message, 311 to 312

access control

- administrative rights, 68
- changing, 283
- changing access rights, 19
- checking, 282 to 283
- NIS+ access rights, 67 to 68
- Solaris security, 54 to 55

access requests by clients, 31 to 32

access rights, 109 to 135

See also authorization; permissions;
security

- administrative rights, 68
- authorization classes, 110, 111 to 112
- changing, 112 to 113
 - group of objects or entries, 134 to 135
 - object and entry access rights, 128 to 130
 - ownership of objects or entries, 132 to 134

concatenation, 111 to 112

defaults, 112, 119

- changing, 112 to 113, 126 to 127
- displaying values, 126
- overriding, 112
- resetting, 127
- setting, 125 to 127

displaying, 118 to 119

- NIS+ defaults, 124 to 125

introduction, 110 to 120

levels (directory, table, column and entry)

- examples, 114 to 115

- security, 113 to 114

- types of rights and objects acted upon, 116 to 117

niscat command requirements, 176

nisgrpadm command

- requirements, 177

passwd command requirements, 148

removing

- for columns, 132

- for objects or entries, 129

specifying in commands, 112, 120 to 123

- column access rights, 130 to 132

- D option, 112, 124, 125, 127 to 128

- default security values, 125 to 127

- nondefault values at creation time, 127 to 128

storage location, 117

syntax

- class, 121, 122

- group, 122

- objects, 123

- operator, 121, 122

- owner, 122

- rights, 122

- table entries, 123

types, 67 to 68, 110 to 111

- levels and, 116 to 117

addresses

See also IP addresses

name service overview, 2 to 8

NIS maps, 10 to 11

NIS+ root reference, 285 to 286

physical addresses vs. descriptive names, 4 to 8

administration

- access rights, 109 to 135

-
- administrative rights, 68
 - credential information
 - administration, 96 to 98
 - credential information for administrators, 91
 - directories, 183 to 205
 - federating NIS+ with global naming systems, 285 to 289
 - file system namespace, 291 to 299
 - FNS in NIS+, 253 to 284
 - groups, 173 to 182
 - keys, 99 to 107
 - NIS+ administrator, 68
 - NIS+ directories, 183 to 205
 - NIS+ tables, 207 to 231
 - passwords, 144 to 172
 - printer namespace, 301 to 304
 - removing credential information, 97 to 98
 - aging passwords, 158 to 168
 - aging vs. expiration, 164
 - calculating dates, 152 to 153
 - date of last change, 150
 - described, 158 to 159
 - expiration date for password
 - privilege, 151 to 152, 164 to 166
 - forcing users to change passwords, 159
 - maximum days before changing, 150 to 151, 160, 170
 - maximum days between logins, 151, 166 to 167
 - minimum days before changing, 150, 161, 170
 - nobody class and, 147
 - operations affected, 159
 - passwd file defaults, 168 to 171
 - MAXWEEKS setting, 170
 - MINWEEKS setting, 170
 - nsswitch.conf file and, 168
 - PASSLENGTH setting, 169, 171
 - principles, 169
 - WARNWEEKS setting, 170
 - setting criteria for multiple users, 168
 - turning off aging, 163
 - turning off privilege expiration, 166
 - warning of required change, 151, 162 to 163, 170
 - world class and, 147
 - aliases
 - host names, 261
 - mail aliases, 11
 - all hosts context, 258, 260
 - all users context, 258, 261 to 262
 - API (application programmer's interface) for NIS+, 20
 - application programmer's interface (API) for NIS+, 20
 - applications, 337
 - asterisk (*)
 - in group member listing, 176
 - regular expression symbol, 220
 - at sign (@) in group member listing, 176
 - attributes
 - adding attributes or values, 276, 278
 - listing, 277
 - managing and examining, 276 to 278
 - modifying values, 278
 - removing attributes or values, 277
 - replacing, 278
 - AUTH_SYS security, 59
 - authentication
 - See also* authorization; credentials
 - credential/authentication process, 74 to 79
 - authentication components, 75
 - credentials vs. credential information, 74 to 75
 - login phase, 75, 76 to 77
 - preparation phase, 75, 76
 - request phase, 76, 77 to 79
 - defined, 12, 17
 - DES credentials, 60 to 61
 - described, 56
 - process, 56 to 57
 - Solaris Secure RPC gate, 55
 - "authentication denied" message, 321
 - "authentication error" message, 321

-
- authorization
 - See also* access rights; authentication; permissions
 - defined, 12, 17
 - described, 56, 63
 - NIS+ authorization classes, 63 to 67, 110
 - access right syntax, 121, 122
 - displaying access rights, 118 to 119
 - group, 63, 65 to 66, 110, 111
 - hierarchy of objects and, 66 to 67
 - nobody, 64, 66, 110, 111
 - owner, 63, 64, 110, 111
 - world, 63, 66, 110, 111
 - passwd command requirements, 148
 - process, 57
 - auto_home table
 - auto_master table and, 404, 405
 - columns, 404
 - described, 46, 404
 - example, 405
 - nsswitch.conf file category, 238
 - auto_master table
 - auto_home table and, 404, 405
 - columns, 405
 - described, 46, 405
 - examples, 405 to 406
 - nsswitch.conf file category, 238
 - automounter, 293 to 294
 - multiple locations, 298
 - troubleshooting, 318
 - variable substitution, 298 to 299
 - automounter maps
 - auto_home table, 46, 404 to 405
 - auto_master table listing, 406
- B**
- B
 - nisinit option, 195, 196
 - rpc.nisd option
 - killing the daemon, 312, 337
 - starting with DNS-forwarding capabilities, 193, 194
- backslash (\)
 - bootparams table input file format, 407
 - regular expression symbol, 220
 - bindings
 - See also* contexts
 - access control
 - changing, 283
 - checking, 282 to 283
 - composite names to references, 273 to 275
 - displaying, 267 to 268
 - NIS vs. NIS+, 316
 - printers, 301
 - removing for composite names, 275
 - renaming, 275
 - blanks in object names, 317
 - bold typeface in this manual, xxxii
 - bootparams file, 406
 - bootparams NIS map, 10
 - bootparams table
 - columns, 406 to 407
 - described, 45, 406
 - example, 407
 - input file format, 407
 - Bourne shell default prompts, xxxii
 - browsing
 - See also* displaying; listing
 - FNS structures using NIS+ commands, 280 to 281
 - “busy try again later” message, 332
- C**
- C
 - nisinit option, 195, 196
 - nisping option
 - checkpoint operations fail consistently, 309
 - checkpointing, 52, 199, 200 to 201
 - performance sluggish when in progress, 333
 - nisupdkeys option, 106
 - c

-
- nisgrep option, 221
 - nisgrpadm option, 177, 178 to 179
 - nisinit option, 195, 196
 - nismatch option, 221
 - nistbladm option, 130 to 131, 209 to 210
 - rpc.nispasswd option, 172
 - c access rights, *See* create access rights
 - C column type, 210
 - C shell
 - default prompts, xxxii
 - nischttl syntax, 205
 - cache manager
 - See also* time-to-live (TTL)
 - described, 196
 - starting, 18, 197
 - stopping, 197
 - cache, directory, *See* directory cache
 - cached public keys problems, 83 to 84
 - “callback” message, 311 to 312
 - “Can’t find suitable transport” message, 315
 - “cannot [do something] with log” message, 337
 - “cannot find” message, 315
 - “cannot get public key” message, 321
 - “Cannot obtain initial context” message, 343
 - case-sensitivity for table columns, 210
 - Changing key message when Adding key message expected, 320
 - checking access control, 282 to 283
 - checkpoint operations
 - checkpointing directories, 199, 200 to 201
 - described, 52
 - disk space problems and, 338
 - fail consistently, 309
 - performance sluggish during, 333
 - child directories, 24
 - chkey command
 - after changing root password, 141
 - changing NIS+ principal keys, 100 to 102
 - creating credential information using dummy password, 93 to 95
 - described, 69
 - re-encrypting existing private key, 100, 101
 - requirements, 101
 - “chkey failed” message, 321
 - clearing
 - See also* removing
 - directory cache, 197
 - keys, 106
 - clients
 - client-server computing, 4
 - cold-start file, 32
 - defined, 4, 31
 - domain access, 31 to 32
 - /etc/.rootkey file preexisting, 331
 - home domain, 31
 - initializing
 - NIS+ clients, 19
 - workstation clients, 194 to 196
 - NIS map for diskless clients, 10
 - NIS+ directories, 22
 - NIS+ tables and, 31
 - overview, 31 to 32
 - removing NIS+, 245 to 246
 - servers as, 36 to 37
 - servers vs., 27
 - closing, *See* stopping
 - cold-start file
 - changing credentials, 326
 - creation, 326
 - credential-related information, 85, 323
 - described, 32
 - initializing clients by, 196
 - updating keys, 106
 - column level
 - access rights
 - adding to existing table, 131 to 132
 - create rights, 117
 - described, 113 to 114

- destroy rights, 117
- example, 114 to 115
- modify rights, 117
- read rights, 116
- removing, 132
- specifying during table
 - creation, 130 to 131
- types and objects acted upon, 116
- authorization classes and, 67
- column types, 210
- searching
 - first column, 221
 - multiple columns, 222
 - particular column, 222
 - regular expressions, 220
- column separators for NIS+ tables, 404
- commands
 - See also specific commands*
 - access right specification, 112, 120 to 123
 - column access rights, 130 to 132
 - default security values, 125 to 127
 - nondefault values at creation time, 127 to 128
 - syntax, 121 to 123
 - browsing FNS structures using NIS+ commands, 280 to 281
 - credential-related commands, 69, 87 to 88
 - FNS context management, 267 to 276
 - group-related commands, 174
 - NIS+ administration commands, 18 to 20
 - NIS+ security (password, credential, and key) commands, 69
 - shell prompts in examples, xxxii
- common key, 77
- compat source for `nsswitch.conf` file, 235, 243
- composite names
 - binding to references, 273 to 275
 - destroying named objects, 276
 - displaying bindings, 267 to 268
 - removing from namespace, 275 to 276
- contexts
 - See also bindings*
 - changing ownership, 283
 - checking naming inconsistencies, 279 to 280
 - creating individually
 - all hosts context, 258, 260
 - all users context, 258, 261 to 262
 - debugging, 258
 - displaying information, 258
 - file context, 266
 - `fncreate` command
 - overview, 257 to 258
 - generic context, 258, 264 to 265
 - namespace identifier
 - context, 258, 266 to 267
 - organization context, 259
 - printer context, 264
 - service context, 263
 - single host context, 260
 - single user context, 262
 - site context, 265
 - debugging creation, 258
 - displaying the binding, 267 to 268
- files
 - administering, 299
 - creating, 294 to 299
 - host-related contexts
 - cannot be created, 345
 - cannot be removed, 346
 - initial context, 343
 - listing contents, 269 to 272
 - managing and examining, 267 to 276
 - mapping to NIS+ objects, 280
 - printers, 302 to 304
 - user-related contexts
 - cannot be created, 345
 - cannot be removed, 346
- continue switch action option, 236
- corrupted credentials, 319, 327 to 328
- corrupted files
 - database or log file, 311 to 312
 - `/var/nis/data` directory, 316
- Courier typeface in this manual, xxxii

- create access rights
 - See also* access rights
 - described, 67 to 68, 111
 - levels and, 116 to 117
 - syntax, 122
- cred table, 85 to 86
 - adding credentials, 18
 - changing keys, 100 to 102
 - checking credentials, 319
 - columns, 86, 408
 - credential-related information, 84, 323
 - DES credential information, 61
 - described, 45, 85, 407
 - linking, 408
 - missing entry, 329
 - `passwd` command and keys, 148
- credentials, 73 to 98
 - See also* authentication; security
 - administering credential information, 96 to 98
 - authentication process, 74 to 79
 - authentication components, 75
 - credentials vs. credential information, 74 to 75
 - login phase, 75, 76 to 77
 - preparation phase, 75, 76
 - request phase, 76, 77 to 79
 - changing DES credentials, 326 to 327
 - checking, 319
 - commands, 69, 87 to 88
 - corrupted or missing, 319, 327 to 328
 - creating, 18
 - creating information, 86 to 96
 - administrator information, 91
 - dummy password example, 93 to 95
 - methods, 86 to 87
 - NIS+ principal information, 91 to 96
 - other domain example, 95
 - preparation phase, 76
 - principal name syntax, 91
 - Secure RPC netname syntax, 90
 - user principal example, 93
 - using `nisaddcred`, 87 to 90
 - workstation example, 96
 - credential information vs., 74 to 75
 - DES credentials, 79 to 84
 - authentication process, 77 to 79
 - changing, 326 to 327
 - contents, 79
 - creating, 80 to 82
 - cred table columns, 86
 - cred table for information, 61
 - described, 60 to 61
 - NIS+ credentials vs., 60
 - `nisaddcred` creation of information, 89 to 90
 - Secure RPC netname, 79
 - verification field, 80
 - displaying information, 88
 - LOCAL credentials
 - cred table columns, 86
 - described, 61 to 62
 - `nisaddcred` creation of information, 89
 - machine credentials, 60
 - NIS+ table, 407 to 408
 - outdated information, 322 to 327
 - overview, 59 to 60
 - `passwd` command and, 147
 - removing, 88
 - removing information, 97 to 98
 - Solaris Secure RPC gate, 55
 - storing information, 84 to 85, 322 to 324
 - updating information, 322 to 324
 - using `nisaddcred`, 87 to 88
 - where information is stored, 84 to 85
 - your own credential information, 97
 - updating public keys, 324 to 327
 - user credentials, 60
 - user types and credential types, 62
 - You *name* do not have Secure RPC credentials error message, 319
- `ctx_dir` directory

- checkpointing, 256
 - creating, 255
 - replicating, 256
 - verifying creation, 255
- D**
- D
 - access right override option, 112, 124, 125, 127 to 128
 - fncreate option, 258
 - nisaddent option, 227
 - nisgrpadm option, 179
 - nisln option, 223
 - passwd option, 148, 155
 - d
 - fnattr option, 277
 - nisaddent option, 227, 231
 - nisdefault option, 124
 - nisgrpadm option, 177, 179
 - nisls option, 185
 - nistbladm option, 211
 - d access rights, *See* destroy access rights
 - dash (-)
 - See also* minus sign (-)
 - netgroup table input file format, 413
 - Data Encryption Standard credentials, *See* DES credentials
 - data.dict file, caution against renaming, 22
 - database corrupted (NIS+), 312
 - date of last password change
 - calculating, 152 to 153
 - displaying, 154 to 155
 - setting, 150
 - debugging context creation, 258
 - defaults
 - access rights
 - changing, 112 to 113
 - specifying, 112, 119
 - displaying using nisdefaults command, 124 to 125
 - listing for NIS+ objects, 19
 - nsswitch.conf search criteria, 236 to 237
 - nsswitch.conf template file, 241
 - password settings, 168 to 172
 - security level, 59
 - setting security values, 125 to 127
 - deleting, *See* removing
 - DES credentials, 79 to 84
 - authentication process, 77 to 79
 - changing, 326 to 327
 - contents, 79
 - creating, 80 to 82
 - cred table columns, 86
 - cred table for information, 61
 - described, 60 to 61
 - NIS+ credentials vs., 60
 - nisaddcred creation of information, 89 to 90
 - Secure RPC netname, 79
 - verification field, 80
 - DES key, 77, 81
 - destroy access rights
 - See also* access rights
 - described, 67 to 68, 111
 - levels and, 116, 117
 - syntax, 122
 - destroying, *See* removing
 - directories, 183 to 205
 - See also* files and file systems
 - access rights granted by servers, 120
 - adding replica to existing directory, 187, 189 to 190
 - automounter, 293 to 294
 - cannot delete org_dir or groups_dir, 311
 - checking for public keys, 88
 - checkpointing, 199, 200 to 201
 - creating, 19
 - on parent server, 188
 - using nismkdir command, 187 to 188
 - creation by fncreate command, 257
 - credential-related information, 84, 323
 - disassociating replicas, 191

-
- expanding into NIS+ domain, 225
 - expanding into NIS-compatible domain, 225
 - fully qualified names, 40
 - home directory contains uppercase letters, 334
 - listing contents, 19
 - terse listing, 185 to 186
 - using `nislsl` command, 184 to 187
 - verbose listing, 186 to 187
 - listing object properties, 184
 - namespace, 24 to 25
 - removing, 19
 - UNIX directories vs., 23, 24
 - NIS+, 22
 - hierarchy and authorization classes, 66 to 67
 - master server, 255
 - `nis_cachemgr` command, 196 to 197
 - `niscat` command, 184
 - `nischttl` command, 203 to 205
 - `nisinit` command, 194 to 196
 - `nislog` command, 201 to 202
 - `nislsl` command, 184 to 187
 - `nismkdir` command, 187 to 190
 - `nisping` command, 198 to 201
 - `nisrmdir` command, 190 to 191
 - `nisshowcache` command, 197 to 198
 - propagating into client files, 326
 - public key propagation, 325
 - removing
 - namespace, 19
 - using `nisrmdir` command, 190 to 191
 - `rpc.nisd` command, 192 to 194
 - server connection to, 27
 - Solaris permissions matrix, 55
 - directory cache
 - See also* time-to-live (TTL)
 - cache manager
 - described, 196
 - starting, 18, 197
 - stopping, 197
 - cached public keys problems, 83 to 84
 - changing credentials, 326
 - clearing, 197
 - credential-related information, 84, 323
 - directory object accumulation by, 34 to 35
 - displaying contents, 197 to 198
 - example, 33
 - initializing, 33, 326
 - listing contents, 19
 - overview, 32 to 35
 - directory level
 - access rights
 - create rights, 116
 - destroy rights, 117
 - modify rights, 117
 - read rights, 116
 - types and objects acted upon, 116
 - authorization classes and, 66
 - directory objects, *See* directories
 - disk space requirements for FNS on NIS+, 254
 - disk space, insufficient, 310, 338
 - diskless workstations
 - See also* workstations
 - configuration information table, 406 to 407
 - described, 406
 - NIS map, 10
 - displaying
 - See also* listing; read access rights
 - access control, 282 to 283
 - access rights, 118 to 119
 - NIS+ defaults, 124 to 125
 - bindings, 267 to 268
 - context creation information, 258
 - credential information, 88
 - defaults using `nisdefaults` command, 124 to 125
 - directory cache contents, 197 to 198
 - NIS+ table contents, 18
 - `NIS_DEFAULTS` variable values, 126
 - password information, 154 to 155
 - size of application or process, 337

- time of last update for server, 198, 199
 - time-to-live values, 203
 - transaction log contents, 201 to 202
 - transaction logs (NIS+), 310
 - DNS and Bind*, xxxi
 - DNS maps, `nsswitch.conf` specification of sources, 404
 - `dns` source for `nsswitch.conf` file, 235
 - DNS table, editing, 286 to 287
 - DNS, *See* Domain Name System (DNS)
 - dollar sign (\$)
 - `NIS_PATH` variable use, 44
 - shell prompts, xxxii
 - Domain Name System (DNS)
 - See also* name services
 - domains defined, 8
 - federating NIS+ under, 286 to 287
 - name service overview, 2 to 8
 - NIS with, 9
 - NIS+ differences, 15
 - NIS-compatibility mode and DNS-forwarding, 18
 - starting the NIS+ daemon, 193, 194
 - `nsswitch.conf` file and DNS forwarding, 241 to 242
 - overview, 8
 - resolver, 8
 - domain names (NIS+)
 - dot (.) ending, 315
 - troubleshooting, 310 to 311
 - warning against changing, 329
 - domains
 - adding replica servers, 326
 - changing machine to different domain, 329
 - client access, 31 to 32
 - creating credential information in another domain, 95
 - described, 25
 - DNS domains, 8
 - expanding directories into NIS+ domain, 225
 - expanding directories into NIS-compatible domain, 225
 - fully qualified names, 40
 - home domain
 - described, 31
 - servers, 36 to 37
 - hosts with same name, 42
 - name changing, 329
 - namespace structure, 25 to 26
 - NIS vs. DNS, 9
 - NIS+ vs. DNS, 15
 - `passwd` command and other domains, 148
 - removing groups and, 179
 - Secure RPC netnames, 80
 - server connection to, 27
 - dots (.)
 - ending domain names, 315
 - ending NIS+ object names, 24
 - `/etc/auto*` table names, 318
 - host names and, 41 to 42
 - NIS+ namespace object names, 23
 - NIS+ principal names vs. Secure RPC netnames, 42
 - regular expression symbols, 220
 - dummy password for creating credential information, 93 to 95
 - dumping NIS+ table information to a file, 231
- E**
- editing the DNS table, 286 to 287
 - encryption, *See* keys
 - ending, *See* stopping
 - entry level
 - access rights
 - adding rights, 128 to 129
 - create rights, 117
 - described, 114
 - destroy rights, 117
 - example, 114 to 115
 - modify rights, 117
 - read rights, 116
 - removing rights, 129

-
- syntax, 123
 - types and objects acted upon, 116
 - adding an entry to a table, 211 to 213
 - authorization classes and, 67
 - changing groups, 135
 - changing ownership, 133
 - changing time-to-live, 205
 - modifying entries, 213 to 214
 - removing entries
 - multiple entries, 215
 - single entry, 214
 - equals sign (=) as access rights operator, 121, 122
 - erasing, *See* removing
 - error messages, 284
 - See also* troubleshooting (FNS); troubleshooting (NIS+); *specific messages or clauses of messages*
 - alphabetic listing, 349 to 402
 - alphabetization rules in this manual, 351
 - context, 349 to 350
 - during login attempt, 139 to 140
 - during password change attempt, 141 to 142
 - name service switch status messages, 236
 - default responses, 237
 - object not found problems, 314 to 318
 - automounter cannot be used, 318
 - blanks in name, 317
 - domain levels not correctly specified, 315
 - files missing or corrupt, 316
 - incorrect path, 315
 - object does not exist, 316
 - replica lagging or out-of-sync, 316
 - symptoms, 315
 - syntax or spelling error, 315
 - RPC error messages, 350
 - severity threshold level, 349
 - status codes, 349
 - tracing cause, 349 to 350
 - /etc files, adding information to NIS+ tables, 18
 - /etc tables, `nsswitch.conf` specification of sources, 404
 - /etc/bootparams file, 406
 - /etc/defaults/passwd file, 168 to 171
 - MAXWEEKS setting, 170
 - MINWEEKS setting, 170
 - `nsswitch.conf` file and, 168
 - PASSLENGTH setting, 169, 171
 - principles, 169
 - WARNWEEKS setting, 170
 - /etc/group file, adding group table contents, 410
 - /etc/nsswitch.conf file
 - auto_home and auto_master table information, 238
 - comments in, 238
 - compatibility with +/- syntax, 235, 242 to 243
 - described, 12
 - DNS-forwarding control, 241 to 242
 - fails to perform correctly, 314
 - files pointer, 168 to 171
 - format, 234 to 238
 - overriding with `passwd -r` command, 146
 - overview, 233 to 234
 - `passwd` entry, 144, 145 to 146, 339
 - `passwd_compat: nisplus` entry, 235
 - search criteria
 - defaults, 236 to 237
 - overview, 235
 - switch action options, 236
 - switch status messages, 236
 - sources, 235
 - specifying information sources, 404
 - syntax errors, 238
 - template files
 - default template, 241
 - described, 233 to 234
 - examples, 240 to 241

- overview, 239
- timezone table and, 238
- See also* name service switch
- `/etc/nsswitch.files` file
 - described, 233 to 234, 239
 - example, 241
 - using, 239
- `/etc/nsswitch.nis` file
 - described, 233 to 234, 239
 - example, 240
 - using, 239
- `/etc/nsswitch.nisplus` file
 - described, 233 to 234, 239
 - example, 240
 - using, 239, 241
- `/etc/passwd` file
 - adding passwd table contents, 416
 - changing passwords in NIS+
 - environment, 330
 - credential-related information, 85, 324
 - NIS+ password and login password
 - in, 330
- `/etc/.rootkey` file
 - changing machine to different
 - domain, 329
 - preexisting, 331
- `/etc/syslog.conf` file, severity
 - threshold for error reporting, 349
- Ethernet addresses
 - ethers table, 408 to 409
 - NIS maps, 10
- ethers table
 - columns, 408
 - described, 46, 408
 - Ethernet address form, 409
- `ethers.byaddr` NIS map, 10
- `ethers.byname` NIS map, 10
- examining, *See* browsing; displaying; listing
- executing, *See* starting
- expanding directories
 - into NIS+ domain, 225
 - into NIS-compatible domain, 225

- expiration date for password
 - privilege, 164 to 166
 - aging vs. expiration, 164
 - calculating, 152 to 153
 - displaying, 154 to 155
 - setting, 151 to 152, 164 to 165
 - turning off privilege expiration, 166
- expired password, 139 to 140, 322
- explicit group members, 174, 175
- explicit group nonmembers, 175
- export command and `NIS_DEFAULTS`
 - values, 126 to 127

F

- f
 - `fncreate` option
 - all hosts context, 260
 - all users context, 262
 - overview, 258
 - `nisaddent` option, 228
 - `nismrm` option, 192
 - `passwd` option, 159
- F, `rpc.nisd` option, 193
- Federated Naming Service
 - See also* name services
 - browsing FNS structures using NIS+
 - commands, 280 to 281
 - described, 8
 - error messages, 284
 - name service overview, 2 to 8
 - setting up
 - FNS namespace setup, 255 to 256
 - NIS+ service setup, 254 to 255
 - replicating FNS service, 256
 - resource requirements, 254
 - troubleshooting, 284, 343 to 348
 - context cannot be removed by
 - creator, 346
 - `fndestroy`/`fnunbind` does not
 - return “operation failed”, 347 to 348
 - `fnlist` does not list
 - suborganizations, 345

-
- host-related contexts cannot be created, 345
 - initial context cannot be obtained, 343
 - initial context contains nothing, 343
 - “name in use” messages, 346 to 347
 - “no permission” messages, 344
 - user-related contexts cannot be created, 345
- federating
- NIS+ with global naming systems, 285 to 289
 - obtaining NIS+ root reference, 285 to 286
 - under DNS, 286 to 287
 - under X.500, 288 to 289
- files and file systems
- See also* directories; *specific files*
- contexts
 - creating, 266
 - ownership, 266
 - dumping NIS+ table information to a file, 231
 - input file for `fncreate_fs(1M)`
 - alternate format, 299
 - creating, 295 to 297
 - loading NIS+ tables from text files, 227, 228 to 229
 - namespace administration, 291 to 299
 - automounter, 293 to 294
 - file context administration, 299
 - file context creation, 294 to 299
 - NFS file servers, 292 to 293
 - overview, 291 to 294
 - NFS file servers, 292 to 293
 - NIS+ files and directories, 22
 - NIS+ namespace compared to UNIX file system, 23 to 24
 - printer context administration, 302
 - Solaris permissions matrix, 55
- files source for `nsswitch.conf` file, 235
- `fnattr` command
- adding attributes, 276
 - listing attributes, 277
 - modifying attribute values, 278
 - other options, 278
 - overview, 276
 - removing attributes, 277
- `fnbind` command
- binding composite names to references, 273 to 274
 - options, 273
 - syntax, 267
- `fnbind -r` command
- binding composite names to references, 274 to 275
 - options, 273
 - syntax, 267
- `fnbind -s` command, “name in use” message, 347
- `fncheck` command, 279 to 280
- `fncreate` command
- all hosts context creation, 258, 260
 - all users context creation, 258, 261 to 262
 - debugging context creation, 258
 - file context creation, 266
 - FNS namespace setup, 255 to 256
 - generic context creation, 258, 264 to 265
 - host-related contexts cannot be created, 345
 - namespace identifier context creation, 258, 266 to 267
 - NIS_GROUP environment variable setting, 255
 - options, 258
 - organization context creation, 259
 - overview, 257 to 258
 - printer context creation, 264
 - service context creation, 263
 - single host context creation, 260
 - single user context creation, 262
 - site context creation, 265
 - syntax, 257
 - t option, 345

-
- user-related contexts cannot be created, 345
 - `fncreate -s` command
 - “name in use” message, 347
 - `fncreate_fs(1M)` command
 - command-line input, 297 to 299
 - input file
 - alternate format, 299
 - creating, 295 to 297
 - multiple locations, 298
 - options, 294
 - overview, 294 to 295
 - syntax, 294
 - variable substitution, 298 to 299
 - `fndestroy` command
 - context cannot be removed, 346
 - destroying named objects, 276
 - “operation failed” not returned, 347 to 348
 - syntax, 267
 - `fnlist` command
 - displays nothing in initial context, 343
 - listing context contents, 269 to 272
 - options, 269
 - suborganizations not listed, 345
 - syntax, 267
 - `fnlookup` command
 - displaying bindings, 267 to 268
 - options, 267
 - syntax, 267
 - `fnrename` command
 - renaming existing bindings, 275
 - syntax, 267
 - FNS, *See* Federated Naming Service
 - `fns_hosts` table, 281
 - `fns_rserver` creation, 256
 - `fnunbind` command
 - “name in use” message, 346
 - “operation failed” not returned, 347 to 348
 - removing composite names, 275
 - syntax, 267
 - forgotten password, 339
 - `fs` context type, 266
 - See also* files and file systems
 - `ftp` command and password aging, 159
 - fully qualified names
 - characters acceptable in, 42
 - dot (.) ending, 24, 315
 - group names, 40 to 41
 - host names, 41 to 42
 - indexed names, 41
 - name-expansion facility, 43
 - NIS+ domain names, 40
 - NIS+ principal names, 42
 - NIS+ server host names, 336 to 337
 - NIS+ table names, 40 to 41
 - NIS+ table search paths, 48 to 50
 - overview, 37 to 39
- ## G
- `-g`
 - `nisdefault` option, 124
 - `nisls` option, 185
 - `g` access rights, 121
 - generic context creation, 258, 264 to 265
 - generic context type, 264 to 265
 - Generic system error message, 310
 - GIDs, NIS maps, 10, 11
 - glossary, 419 to 428
 - group authorization class
 - access rights, 111
 - displaying for objects, 118 to 119
 - syntax, 122
 - described, 63, 65 to 66
 - group file, adding group table contents, 410
 - group IDs, NIS maps, 10, 11
 - group level and authorization classes, 66
 - group table
 - adding contents to `/etc/group` files, 410
 - columns, 410
 - described, 46, 409
 - `group.bygid` NIS map, 10
 - `group.byname` NIS map, 10

-
- groups
 - adding members, 179 to 180
 - administration, 173 to 182
 - authorization class, 63, 65 to 66
 - cannot add user, 309
 - changing
 - entry's group, 135
 - object's group, 134
 - creating, 178 to 179
 - described, 173
 - displaying directory's group
 - owner, 185
 - fully qualified names, 40 to 41
 - group table, 409 to 410
 - listing members, 180 to 181
 - listing object properties, 175 to 176
 - member types, 174
 - netgroup NIS maps, 11
 - netgroup table, 411 to 413
 - NIS maps, 10
 - niscat command usage, 175 to 176
 - nisgrpadm command, 19, 177 to 182
 - related commands, 174
 - nonmember types, 175
 - removing, 179
 - removing members, 181
 - specifying member types, 174 to 175
 - testing principals for
 - membership, 182
 - groups_dir directory
 - cannot be deleted, 311
 - creating, 19
 - described, 24, 65
 - fully qualified names, 40 to 41
- H**
- H
 - nisinit option, 195
 - nisupdkeys option, 106
 - h
 - niscat option, 216
 - nisdefault option, 124
 - nisgrep option, 221
 - nislog option, 201
 - nismatch option, 221
 - hard disk space requirements for FNS on NIS+, 254
 - hard disk space, insufficient, 310, 338
 - hash character (#)
 - nsswitch.conf file comments, 238
 - shell prompts, xxxii
 - home directory, uppercase letters in, 334
 - home domain
 - described, 31
 - servers, 36 to 37
 - /home mount point, 404
 - host context type, 260
 - host name
 - initializing clients by, 195
 - NIS+ queries hang after
 - changing, 336 to 337
 - hostname context type, 260
 - hosts
 - aliases, 261
 - context creation
 - all hosts, 258, 260
 - single host, 260
 - context ownership, 260, 261
 - domains with same name, 42
 - fully qualified names, 41 to 42
 - NIS maps, 10
 - hosts table
 - columns, 410
 - described, 45, 410
 - hosts.byaddr NIS map, 10
 - hosts.byname NIS map, 10
 - hung system, *See* system hang problems
 - hyphen, *See* dash (-); minus sign (-)
- I**
- i
 - nis_cachemgr option, 197
 - nism option, 192
 - I column type, 210
 - identifier of Secure RPC netnames, 80

identifiers, namespace, context
 creation, 258, 266 to 267

illegal object problems, 308 to 309

Illegal object type message, 308

implicit group members, 174, 175

implicit group nonmembers, 175

in.named daemon, 8

inactivity maximum, 339

inconsistencies in context naming,
 checking, 279 to 280

indexed names, 41, 123

initial context, 343
 See also contexts

initializing
 directory cache, 33, 326
 NIS+ client or server, 19
 reinitializing NIS+ clients, 329
 root master server, 195, 196
 server or client with preexisting
 /etc/.rootkey file, 331
 workstation clients, 194 to 196

input file for `fncreate_fs(1M)`
 alternate format, 299
 creating, 295 to 297

input file format requirements, 404

installation (NIS+), server slow at startup
 after, 335

“insufficient permission” message, 318,
 321, 331

Internet
 See also Domain Name System (DNS)
 Ethernet addresses table, 408 to 409
 network masks table for
 subnetting, 413
 networks table, 414
 NIS connection, 9
 NIS map of services, 11
 NIS+ connection, 16
 protocols table, 416
 services table, 417

“invalid principal name” message, 313

IP addresses
 See also addresses

hosts table, 410

netmasks table, 413

updating, 107

italic typeface in this manual, xxxii

K

keylogin command
 authentication, 57
 creating DES credentials and, 81
 described, 69, 76
 Secure RPC password different from
 login password, 76, 82 to 83,
 99 to 100, 330 to 331

keylogin process, 99 to 100

keylogout command, 69, 76 to 77

keys
 See also public keys; private keys
 administration, 99 to 107
 authentication process, 77 to 79
 cached public keys problems, 83 to 84
 changing keys
 chkey command, 100 to 102
 nonroot server keys, 105
 root keys from another
 machine, 104
 root keys from root, 102
 root replica keys from the
 replica, 104 to 105

clearing, 106

commands, 69

common key, 77

creation by `nisaddcred`, 89

DES key (random key or random DES
 key), 77

NIS map, 11

`passwd` command and, 148

re-encrypting existing private
 key, 100, 101

Secure RPC password different from
 login password, 76, 82 to 83,
 99 to 100, 330 to 331

updating public keys, 20, 105 to 107,
 324 to 327

examples, 107

- nisupdkeys command
 - arguments, 106
 - nisupdkeys command
 - overview, 105 to 106
 - updating IP addresses, 107
 - where information is stored, 84 to 85, 323
 - keyserver command, 69
 - keyserver daemon failure, 328
 - “keyserver fails to encrypt” message, 321
 - keyserver
 - credential-related information, 84, 323
 - keyserver daemon failure, 328
 - killing, *See* stopping
 - Korn shell default prompts, xxxii
- L**
- L
 - fnbind option, 273, 274
 - fnlookup option, 267
 - nischttl option, 204 to 205
 - nisl option, 223
 - nisl option, 185
- l
 - fnattr option, 277
 - fnlist option, 269, 271 to 272
 - nisgrpadm option, 177, 180 to 181, 182
 - nisl option, 185, 186 to 187
 - passwd option, 158
- launching, *See* starting
- less than *N* days since the last change message, 141, 161
- levels of access
 - examples, 114 to 115
 - security, 113 to 114
 - types of access rights and, 116 to 117
- levels of security
 - described, 58 to 59
 - password commands and, 59
 - servers running at level 0, 319
 - setting for NIS+ daemon, 193
- linking NIS+ tables, cred table, 408
- links (XFN)
 - creating and binding, 273, 274
 - displaying binding, 267
- links, symbolic, *See* symbolic links
- listing
 - See also* displaying; read access rights
 - attributes and their values, 277
 - browsing FNS structures using NIS+ commands, 280 to 281
 - context contents, 269 to 272
 - context naming inconsistencies, 279 to 280
 - directory contents, 184 to 187
 - nisl command options, 185
 - terse listing, 185 to 186
 - verbose listing, 186 to 187
 - directory object properties, 184
 - fnlist does not list
 - suborganizations, 345
 - fns_hosts table contents, 281
 - group members, 180 to 181
 - group object properties, 175 to 176
 - initial context contains nothing, 343
 - NIS+ directory contents, 19
 - NIS+ object defaults, 19
 - NIS+ objects used by FNS, 280 to 281
 - NIS+ shared cache contents, 19
 - printing references used for
 - binding, 273, 274
 - suborganizations using nisl, 345
 - table contents, 217
 - table object properties, 217 to 218
- loading NIS+ tables, 226 to 231
 - methods of populating tables, 51 to 52
 - replacing, merging, or
 - appending, 226
 - using NIS maps, 227, 229 to 231
 - using text files, 227, 228 to 229
- LOCAL credentials
 - cred table columns, 86
 - described, 61 to 62
 - nisaddcred creation of
 - information, 89

locked password, 322

locking passwords, 157

- unlocking, 158

log files

- checkpointing, 52
- corrupted, 311 to 312
- naming convention, 52
- slow startup and, 334
- too large, 309, 334, 338
- transaction log
 - cannot be truncated, 310
 - described, 29
 - displaying contents, 201 to 202
 - replica updating process, 28 to 30

logging in

- authentication process, 75, 76 to 77
- maximum days between logins, 151
- maximum login time period, 172
- process, 138
- remote logins
 - netgroup table, 411 to 413
 - user cannot remote log in to
 - remote domain, 340
- Solaris security, 54 to 55
- troubleshooting
 - Login incorrect
 - message, 139, 157, 166, 321
 - NIS+ password and login
 - password in
 - /etc/passwd file, 330
 - password expired
 - message, 139 to 140
 - password will expire
 - message, 140, 162
 - Permission denied
 - message, 140
 - Secure RPC password different
 - from login password, 76, 82 to 83, 99 to 100, 330 to 331
 - Too many failures - try
 - later message, 139
 - Too many tries; try again
 - later message, 139
 - user cannot log in, 339
 - user cannot log in after password
 - change, 313 to 314, 340
 - user cannot remote log in to
 - remote domain, 340
 - user login same as machine
 - name, 319 to 320
 - logging out, private key security, 76 to 77
 - login command
 - password aging and, 159
 - Secure RPC password different from
 - login password, 99 to 100
- Login incorrect message, 139, 157, 166, 321

M

-M

- nisaddent option, 227
- niscat option, 216
- nisgrep option, 221
- nislsl option, 185
- nismatch option, 221
- rpc.nisd option, 334

-m

- fnattr option, 278
- nisaddent option, 226, 227, 228, 229, 230
- nislsl option, 185
- nismkdir option, 187
- nistbladm option, 213 to 214

m access rights, *See* modify access rights

machine credentials, 60, 62

See also credentials

machine domain names

- dot (.) ending, 315

machines

- changing domains, 329
- /etc/.rootkey file preexisting, 331
- reinitializing NIS+ clients, 329
- root password change causes
 - problem, 332

mail aliases, 11

mail aliases table, 46, 411

mail.aliases NIS map, 11
 mail.byaddr NIS map, 11
Managing NFS and NIS, xxxi
 mapping FNS contexts to NIS+
 objects, 280
 maps (DNS), nsswitch.conf
 specification of sources, 404
 maps (NIS)
 described, 10
 loading NIS+ tables from, 18, 51, 229
 to 231
 NIS+ table types matching maps, 229
 to 231
 NIS+ tables vs., 16
 nsswitch.conf specification of
 sources, 404
 table of, 10 to 11
 Master server busy, full dump
 rescheduled message, 341
 master servers
 See also servers
 defined, 9, 12
 displaying time of last update, 198,
 199
 listing directory contents from, 185
 listing table entries from, 216, 221
 NIS, 9
 NIS+, 12, 15
 replicas and, 28
 root master server
 defined, 28
 initializing, 195, 196
 MAXWEEKS default for passwords, 170
 memory, insufficient, 337
 messages, *See* error messages
 minus sign (-)
 See also dash (-)
 access rights operator, 121, 122
 group names, 175
 nsswitch.conf compatibility with
 +/- syntax, 235, 242 to 243
 MINWEEKS default for passwords, 170
 modify access rights
 See also access rights

 described, 67 to 68, 110
 levels and, 116, 117
 syntax, 122
 mount points
 auto_home table, 404
 auto_master table, 405
 mounting directories, 293 to 294
 mounts, remote, netgroup table, 411 to
 413

N

n access rights, 121
 “name in use” messages
 fnbind -s command, 347
 fncreate -s command, 347
 fnunbind command, 346
 name service switch, 14, 233 to 243
 See also nsswitch.conf file
 action options, 236
 described, 17, 233
 overview, 233 to 234
 status messages, 236
 name services
 See also Domain Name System (DNS);
 Federated Naming Service;
 NIS; NIS+
 advantages, 4 to 8
 overview, 2 to 8
 for printer context
 administration, 302 to 304
 names, *See* composite names; domain
 names (NIS+); fully qualified
 names
 namespace, 21 to 44
 See also fully qualified names
 administration problems, 308 to 311
 cannot add user to group, 309
 cannot delete org_dir or
 groups_dir, 311
 cannot truncate transaction log
 file, 310
 checkpoint fails consistently, 309
 disk space insufficient, 310

- domain name confusion, 310 to 311
- illegal object problems, 308 to 309
- logs grow too large, 309
- nisinit fails, 309
- symptoms, 308
- database problems, 311 to 312
 - multiple `rpc.nisd` processes, 311 to 312
 - symptoms, 311
- directories, 24 to 25
- DNS namespace, 8
- domain structure, 25 to 26
- naming conventions, 37 to 43
 - characters acceptable, 42
 - directory object names, 40
 - host names, 41 to 42
 - name-expansion facility, 43
 - NIS+ domain names, 40
 - NIS+ principal names, 42
 - overview, 37 to 39
 - table entry names, 41
 - tables and group names, 40 to 41
- NIS+ clients, 31 to 32
- NIS+ principals, 30
- NIS_PATH variable
 - overview, 43 to 44
 - performance and, 44
- performance problems, 335
- removing
 - NIS+ directories and replicas, 19
 - NIS+ namespace, 248 to 250
 - NIS+ objects, 19
- servers, 27 to 30
- servers as clients, 36 to 37
- structure of NIS+ namespace, 22 to 24
- UNIX file system compared to, 23 to 24
- updating process, 28 to 30
- namespace identifiers
 - context creation, 258, 266 to 267
 - context ownership, 267
- naming inconsistencies, checking for contexts, 279 to 280
- netgroup NIS map, 11
- netgroup table
 - columns, 412
 - described, 46, 411
 - examples, 412
 - input file format, 412
- netgroup.byhost NIS map, 11
- netgroup.byuser NIS map, 11
- netgroups
 - group table, 409 to 410
 - NIS maps, 11
- netid.byname NIS map, 11
- netmasks table
 - columns, 413
 - described, 46, 413
 - example, 413
- netmasks.byaddr NIS map, 11
- netnames (Secure RPC)
 - creation by `nisaddcred`, 89
 - DES credentials, 80
 - syntax, 90
- Network Information Control Center (NIC), 414
- network information services, *See* Domain Name System (DNS); Federated Naming Service; name services; NIS; NIS+
- network masks, NIS maps, 11
- network numbers and name service names, 4 to 5
- networks table, 46, 414
- networks.byaddr NIS map, 11
- networks.byname NIS map, 11
- newkey command, 69
- NIC (Network Information Control Center), 414
- NIS
 - See also* name services
 - DNS with, 9
 - DNS-forwarding control, 242
 - Internet connection, 9
 - maps
 - described, 10

-
- loading NIS+ tables from, 18, 51, 229 to 231
 - NIS+ table types matching
 - maps, 229 to 230
 - NIS+ tables vs., 16
 - `nsswitch.conf` specification of sources, 404
 - table of, 10 to 11
 - name service overview, 2 to 8
 - NIS+ compatibility problems, 312 to 314
 - `nsswitch.conf` file fails to perform correctly, 314
 - symptoms, 312 to 313
 - user cannot log in after password change, 313 to 314
 - NIS+ vs.
 - advantages of NIS+, 12 to 13
 - differences, 13 to 16
 - NIS-compatibility mode of NIS+, 17 to 18
 - DNS-forwarding and, 18, 193, 194
 - expanding directories into NIS-compatible domain, 225
 - security issues, 18, 55
 - starting the NIS+ daemon, 193 to 194
 - `nsswitch.conf` compatibility with `+/- syntax`, 235, 242 to 243
 - `nsswitch.nis` file, described, 233 to 234
 - overview, 8 to 11
 - printer context administration, 302
- NIS maps
- `nsswitch.conf` specification of sources, 404
- `nis` source for `nsswitch.conf` file, 235
- NIS+, 11 to 20
- See also* name services; NIS+ tables; security
 - administration commands, 18 to 20
 - application programmer's interface (API), 20
 - changing passwords, 330
 - checking if running, 341
- daemon
- credential-related
 - information, 84, 323
 - starting, 193 to 194
 - stopping, 194
- described, 11 to 12
- DNS differences, 15
- DNS-forwarding control, 242
- federating with global naming systems, 285 to 289
- obtaining NIS+ root
 - reference, 285 to 286
 - under DNS, 286 to 287
 - under X.500, 288 to 289
- files and directories, 22
- FNS administration, 253 to 284
- access control checking, 282 to 283
 - browsing FNS structures using NIS+ commands, 280 to 281
 - error messages, 284
 - FNS attribute management, 276 to 278
 - FNS context creation, 257 to 267
 - FNS context management, 267 to 276
 - maintaining consistency between NIS+ and FNS, 278 to 280
 - mapping FNS contexts to NIS+ objects, 280
 - replicating FNS service, 256
 - resource requirements, 254
 - setting up FNS namespace, 255 to 256
 - setting up NIS+ service for FNS, 254 to 255
 - troubleshooting, 284
- groups, group table, 409 to 410
- hierarchical structure, 13
- incremental updates, 13, 15
- Internet connection, 16
- name service overview, 2 to 8
- name service switch, described, 17

-
- NIS compatibility problems, 312 to 314
 - nsswitch.conf file fails to perform correctly, 314
 - symptoms, 312 to 313
 - user cannot log in after password change, 313 to 314
 - NIS vs.
 - advantages of NIS+, 12 to 13
 - differences, 13 to 16
 - NIS-compatibility mode, 17 to 18
 - DNS-forwarding and, 18, 193, 194
 - expanding directories into NIS-compatible domain, 225
 - security issues, 18, 55
 - starting the NIS+ daemon, 193 to 194
 - nsswitch.conf compatibility with +/- syntax, 235, 242 to 243
 - nsswitch.nisplus file
 - described, 233 to 234
 - overview, 11 to 20
 - printer context administration, 303 to 304
 - removing, 245 to 250
 - NIS+ from client machine, 245 to 246
 - NIS+ from server, 246 to 248
 - NIS+ namespace, 248 to 250
 - root reference, 285 to 286
 - security
 - See also* access rights; credentials; passwords; permissions
 - access rights, 67 to 68
 - administrator, 68
 - authorization classes, 63 to 67, 110
 - commands, 69
 - credentials and authentication, 59 to 62
 - NIS vs. NIS+, 13
 - overview, 56 to 58
 - password commands, 59
 - principals, 58
 - security levels, 58 to 59
 - Solaris 1.x and, 17 to 18
 - tables, *See* NIS+ tables
 - troubleshooting, 307 to 348
 - miscellaneous problems, 341 to 342
 - namespace administration problems, 308 to 311
 - namespace database problems, 311 to 312
 - NIS compatibility problems, 312 to 314
 - object not found problems, 314 to 318
 - ownership and permission problems, 318 to 321
 - security problems, 321 to 332
 - slow performance problems, 332 to 336
 - system hang problems, 332 to 337
 - system resource problems, 337 to 338
 - user problems, 338 to 341
 - NIS+ and DNS Setup and Configuration Guide*, xxxi
 - NIS+ and FNS Administration Guide*
 - intended audience, xxix
 - organization, xxix to xxxi
 - overview, xxix
 - related books, xxxi
 - required knowledge, xxix
 - shell prompts, xxxii
 - typographic conventions, xxxii
 - NIS+ tables, 45 to 52, 403 to 418
 - See also* column level; entry level; nistbladm command; table level; *specific tables*
 - access rights, 113 to 117
 - how servers grant rights, 119 to 120
 - adding entries, 211 to 213
 - adding information from /etc files or NIS maps, 18
 - administration, 207 to 231

- alternate sources of information, 233 to 234
- changing entries, 213 to 214
- changing time-to-live for entries, 203, 205
- clients and, 31
- column requirements, 209
- column separators, 404
- column types, 210
- creating
 - unpopulated tables, 19, 50 to 51
 - using `nistbladm`
 - command, 209 to 210
- creating links, 223
- displaying contents, 18
- dumping table information to a file, 231
- `/etc` files corresponding to, 404
- expanding directories into NIS+ domain, 225
- expanding directories into NIS-compatible domain, 225
- fully qualified names, 40 to 41
- indexed names, 41
- input file format requirements, 404
- listing object properties, 217 to 218
- listing table contents, 217
- loading information, 226 to 231
 - loading from NIS maps, 227, 229 to 231
 - loading from text files, 227, 228 to 229
 - methods of populating tables, 51 to 52
 - replacing, merging, or appending, 226
- NIS maps and matching tables, 229 to 230
- NIS maps vs., 16
- `nisaddent` command, 226 to 231
- `niscat` command, 216 to 218
- `nisgrep` command, 219 to 222
- `nisl` command, 222 to 223
- `nismatch` command, 219 to 222
- `nissetup` command, 224 to 225
- `nistbladm` command, 208 to 216
- `nsswitch.conf` specification of sources, 404
- preconfigured tables, 16, 45 to 46
- removing, 211
- removing entries
 - multiple entries, 215
 - single entry, 214
- searching for entries, 19
 - first column, 221
 - multiple columns, 222
 - particular column, 222
 - regular expressions, 220
- setting up, 50 to 52
- structure, 45 to 50
 - columns and entries, 47 to 48
 - search paths, 48 to 50
- system tables, 16, 45 to 46
 - creating, 50 to 51
- updating process, 52
- NIS+ Transition Guide*, xxxi
- `nis_cachemgr` program, 196 to 197
 - cached public keys problems, 83 to 84
 - described, 18, 196
 - not running, 335
 - starting the cache manager, 18, 197
- `NIS_COLDSTART` file
 - See also* cold-start file
 - changing credentials, 326
 - creation, 326
 - credential-related information, 85, 323
 - initializing clients by, 196
 - updating keys, 106
- `NIS_DEFAULTS` variable
 - access right defaults, 112, 119
 - changing defaults, 112 to 113, 126 to 127
 - displaying values, 126
 - group object properties, 176, 178
 - overriding, 112
 - resetting to original values, 127
 - setting default security values, 125 to 127

`nis_dump_svc`: one replica is already resyncing message, 342
`NIS_DUMPLATER` error code constant, 342
`NIS_GROUP` environment variable, 179, 255
`NIS_PATH` variable
 overview, 43 to 44
 performance and, 44, 333
`NIS_SHARED_DIRCACHE` file
 See also directory cache; time-to-live (TTL)
 changing credentials, 326
 credential-related information, 84, 323
 initializing, 326
 starting the cache manager, 18, 196 to 197
`nisaddcred` command
 Changing key message, 320
 creating credential information, 87 to 88
 administrator information, 91
 dummy password example, 93 to 95
 NIS+ principals' information, 91 to 96
 other domain example, 95
 principal name syntax, 91
 process of creation, 89 to 90
 related commands, 88
 Secure RPC netname syntax, 90
 user principals example, 93
 workstation example, 96
 described, 18, 69
 public key generation, 325
 removing credential information, 97 to 98
 updating your own credential information, 97
`nisaddent` command, 226 to 231
 described, 18
 dumping table contents to a file, 231
 loading data from files, 52, 228 to 229
 loading data from NIS maps, 51, 229 to 231
 overview, 226
 populating tables, 52
 replacing, merging, or appending, 226
 specifying nondefault security values, 127 to 128
 syntax, 227
`niscat` command, 216 to 218
 checking access control, 282 to 283
 checking directories for public keys, 88
 checking permissions, 319
 described, 18, 154, 215
 listing `fns_hosts` contents, 281
 listing object properties
 directories, 184
 groups, 175 to 176
 tables, 217 to 218
 listing table contents, 217
 Server busy. Try again message, 336
 syntax, 216
 viewing access rights, 118 to 119
`nischgrp` command, 283
 changing entry's group, 135
 changing object's group, 134
 described, 19, 174
`nischmod` command, 283
 adding access rights, 128 to 129
 changing context access control, 283
 described, 19
 removing access rights, 129
`nischown` command, 283
 changing context ownership, 283
 changing entry ownership, 133
 changing object ownership, 133
 described, 19
`nischttl` command, 203 to 205
 changing object's time-to-live, 204 to 205
 changing table entry's time-to-live, 205
 described, 19

-
- options, 204
 - overview, 203 to 204
 - `nisclient` command
 - Changing key message, 320
 - cold-start file creation, 326
 - credential information creation, 76, 87
 - removing NIS+ installed by, 245 to 246
 - NIS-compatibility mode, 17 to 18
 - DNS-forwarding and, 18, 193, 194
 - expanding directories into NIS-compatible domain, 225
 - security issues, 18, 55
 - starting the NIS+ daemon, 193 to 194
 - `nisdefaults` command
 - described, 19, 174
 - displaying NIS+ defaults, 124 to 125
 - displaying subsets of values, 125
 - displaying time-to-live values, 203
 - options, 124
 - terse format, 125
 - verbose format, 125
 - `nisgrep` command, 219 to 222
 - checking existence of NIS+ password, 329
 - described, 19, 219
 - `nismatch` compared to, 219
 - regular expressions, 220
 - searching tables
 - first column, 221
 - multiple columns, 222
 - particular column, 222
 - syntax, 220
 - `nisgrpadm` command
 - access rights required, 177
 - adding group members, 179 to 180
 - creating groups, 178 to 179
 - described, 19, 177
 - listing group members, 180 to 181
 - related commands, 174
 - removing group members, 181
 - removing groups, 179
 - specifying group member types, 175
 - syntax, 175, 177
 - testing principals for membership, 182
 - `nisinit` command
 - cold-start file creation, 326
 - described, 19
 - fails, 309
 - initializing clients, 194 to 196
 - initializing root master server, 195, 196
 - `nisl` command, 222 to 223
 - creating a link, 223
 - described, 19, 222
 - syntax, 223
 - `nislog` command, 201 to 202
 - `nisl` command, 184 to 187
 - browsing FNS structures, 280 to 281
 - checking existence of objects, 316
 - checking transaction log, 310
 - described, 19, 174
 - displaying suborganizations, 345
 - listing directory contents
 - terse listing, 185 to 186
 - verbose listing, 186 to 187
 - options, 185
 - verifying `ctx_dir` creation, 255
 - `nismatch` command, 219 to 222
 - checking credentials, 319
 - described, 19, 219
 - displaying credential information, 88
 - `nisgrep` compared to, 219
 - regular expressions, 220
 - searching tables
 - first column, 221
 - multiple columns, 222
 - particular column, 222
 - syntax, 220
 - `nismkdir` command, 187 to 190
 - adding replica to existing directory, 187, 189 to 190
 - creating directories, 187 to 188
 - described, 19
 - master server assignment, 255
 - `org_dir` creation, 51
 - problems running on replica, 189
 - public key propagation, 325

- replicating FNS service, 256
 - specifying nondefault security values, 127 to 128
- nispasswd command
 - described, 19, 154
 - security issues, 59
 - using passwd instead, 140, 144
- nisping -C command
 - checkpoint operations fail consistently, 309
 - checkpointing, 52
 - checkpointing directories, 199, 200 to 201
 - performance sluggish when in progress, 333
- nisping command, 198 to 201, 256
 - checkpointing ctx_dir directory, 256
 - described, 198 to 199
 - displaying time of last update, 198, 199
 - performance sluggish when in progress, 333
 - pinging replicas, 199 to 200
- nisplus source for nsswitch.conf file, 235
- nispopulate script
 - creating credential information, 87
 - populating NIS+ tables, 51 to 52
- nisrm command
 - described, 19, 191
 - options, 192
 - removing nondirectory objects, 192
- nisrmdir command
 - cannot delete org_dir or groups_dir, 311
 - described, 19, 190
 - disassociating replicas from directories, 191
 - removing directories, 190 to 191
- nisserver script, creating NIS+ tables, 51
- nissetup command, 224 to 225
 - creating NIS+ tables, 51
 - described, 19, 174, 224
 - expanding directory into NIS+ domain, 225
 - expanding directory into NIS-compatible domain, 225
 - syntax, 224
- nisshowcache command
 - described, 19
 - displaying directory cache contents, 197 to 198
- nistbladm command, 148 to 153, 208 to 216
 - adding access rights to existing table columns, 131 to 132
 - adding entries to tables, 211 to 213
 - changing entries, 213 to 214
 - creating NIS+ tables, 51, 209 to 210
 - described, 20, 148 to 149, 208
 - passwd command vs., 149
 - password operations, 148 to 153
 - calculating number of days, 152 to 153
 - expiration date for password privilege, 151 to 152, 164 to 166
 - maximum days before changing, 150 to 151
 - maximum days between logins, 151, 166 to 167
 - minimum days before changing, 150
 - setting criteria for multiple users, 168
 - shadow column fields (passwd table) and, 149 to 152
 - turning off password privilege expiration, 166
 - warning of required change, 151
 - populating NIS+ tables, 52
 - removing column access rights, 132
 - removing entries
 - multiple entries, 215
 - single entry, 214
 - removing tables, 211
 - security-related options, 130

- setting column access rights when
 - creating tables, 130 to 131
 - setting nondefault table paths, 333
 - specifying nondefault security
 - values, 127 to 128
 - syntax, 208
 - columnspec* components, 209
 - table search paths, 50
 - turning off password aging, 164
 - `nisupdkeys` command
 - arguments, 106
 - cached public keys problems, 83
 - described, 20, 69
 - examples, 107
 - updating directory objects after key
 - change, 91, 104, 105
 - updating IP addresses, 107
 - updating keys manually, 324 to 327
 - updating public keys, 105 to 107
 - “no memory” message, 337
 - “no permission” messages (FNS), 344
 - “no public key” message, 321
 - nobody authorization class
 - access rights, 111
 - displaying for objects, 118 to 119
 - described, 64, 66
 - password-aging constraints and, 147
 - “permission denied” errors, 82 to 83
 - “not exist” message, 315
 - “not found” message, 315
 - “not responding” message, 332
 - NOTFOUND switch status message, 236
 - default response, 237
 - `nsid` context type, 266 to 267
 - `nsswitch.conf` file
 - See also* name service switch
 - `auto_home` and `auto_master` table
 - information, 238
 - comments in, 238
 - compatibility with `+/-` syntax, 235, 242 to 243
 - described, 12
 - DNS-forwarding control, 241 to 242
 - fails to perform correctly, 314
 - `files` pointer, 168 to 171
 - format, 234 to 238
 - overriding with `passwd -r`
 - command, 146
 - overview, 233 to 234
 - `passwd` entry, 144, 145 to 146, 339
 - `passwd_compat: nisplus`
 - entry, 235
 - search criteria
 - defaults, 236 to 237
 - overview, 235
 - switch action options, 236
 - switch status messages, 236
 - sources, 235
 - compatibility with `+/-`
 - syntax, 235, 242 to 243
 - specifying information sources, 404
 - syntax errors, 238
 - template files
 - default template, 241
 - described, 233 to 234
 - examples, 240 to 241
 - overview, 239
 - timezone table and, 238
 - `nsswitch.files` file
 - described, 233 to 234, 239
 - example, 241
 - using, 239
 - `nsswitch.nis` file
 - described, 233 to 234, 239
 - example, 240
 - using, 239
 - `nsswitch.nisplus` file
 - described, 233 to 234, 239
 - example, 240
 - using, 239, 241
 - number sign (#)
 - `nsswitch.conf` file comments, 238
 - shell prompts, xxxii
- O**
- O
 - `fnattr` option, 278
 - `fnbind` option, 273

-
- o
 - fncreate option
 - all users context, 261
 - organization context, 260
 - overview, 258
 - single user context, 262
 - niscat option
 - listing directory object
 - properties, 184
 - listing group object
 - properties, 175 to 176
 - listing table object
 - properties, 216, 217 to 218
 - o access rights, 121
 - “object problem” messages, 308
 - objects
 - See also* directories
 - access rights
 - adding rights, 128 to 129
 - removing rights, 129
 - syntax, 123
 - administrative rights, 68
 - changing groups, 134
 - changing ownership, 133
 - changing time-to-live, 204 to 205
 - defined, 23
 - destroying named objects, 276
 - hierarchy and authorization
 - classes, 66 to 67
 - listing object properties
 - directories, 184
 - groups, 175 to 176
 - tables, 217 to 218
 - NIS+ objects permissions matrix, 56
 - removing nondirectory objects, 191 to 192
 - request authentication, 76, 77 to 79
 - opening, *See* starting
 - “operation failed” message not returned with
 - fndestroy/fnunbind, 347 to 348
 - operator syntax for access rights, 121
 - org context type, 259
 - org_dir directory
 - cannot be deleted, 311
 - creating, 19, 50 to 51
 - cred table for, 407 to 408
 - described, 24
 - fully qualified names, 40 to 41
 - organizational units
 - contexts
 - creating, 259
 - ownership, 260
 - double slashes in organization names, 283
 - trailing slash in organization names, 284
 - “out of disk space” message, 338
 - owner authorization class
 - access rights, 111
 - displaying for objects, 118 to 119
 - syntax, 122
 - described, 63, 64
 - ownership
 - changing
 - authorization class, 65
 - entry ownership, 133
 - group owner of NIS+ object, 19
 - object ownership, 133
 - owner of NIS+ object, 19
 - displaying directory’s group owner, 185
 - owner authorization class, 63, 64
 - troubleshooting, 318 to 321
 - credentials missing or corrupted, 319, 327 to 328
 - server running at security level 0, 319
 - symptoms, 318
 - user login same as machine name, 319 to 320
- P**
- P
 - nisaddcred option, 91, 92
 - nisaddent option, 227
 - nischtttl option, 204, 205

-
- p
 - chkey option, 100
 - nisaddcred option, 90, 92
 - nisdefault option, 124
 - nistbladm option, 50
 - parent directories, 24
 - partially qualified names
 - defined, 37
 - name-expansion facility, 43
 - PASSLENGTH default for passwords, 171
 - passwd command, 145 to 148
 - access rights required, 148
 - aging passwords, 158 to 168
 - aging vs. expiration, 164
 - calculating dates, 152 to 153
 - date of last change, 150
 - described, 158 to 159
 - forcing users to change passwords, 159
 - maximum days before changing, 160
 - minimum days before changing, 161
 - nobody class and, 147
 - operations affected, 159
 - turning off aging, 163
 - warning of required change, 162 to 163
 - world class and, 147
 - changing passwords
 - another's password, 156
 - choosing passwords, 142 to 143
 - failures, 142
 - forcing users to change passwords, 159
 - root password, 141, 157
 - your password, 140 to 142, 156
 - credentials and, 147
 - described, 20, 69, 145
 - displaying password
 - information, 154 to 155
 - keys and, 148
 - locking passwords, 157
 - NIS+ environment and, 147
 - nispaswd command and, 140, 144
 - nistbladm command vs., 149
 - nsswitch.conf file and, 145 to 146
 - other domains and, 148
 - permissions and, 147 to 148
 - r option, 146
 - root password change causes
 - problem, 332
 - security issues, 59
 - unlocking passwords, 158
 - user cannot change password, 340
 - user cannot log in after password change, 313 to 314, 340
 - yppaswd command and, 145
 - passwd file
 - adding passwd table contents, 416
 - credential-related information, 85, 324
 - defaults file, 168 to 171
 - MAXWEEKS setting, 170
 - MINWEEKS setting, 170
 - nsswitch.conf file and, 168
 - PASSLENGTH setting, 169, 171
 - principles, 169
 - WARNWEEKS setting, 170
 - NIS+ password and login password in, 330
 - passwd map, credential-related information, 85, 324
 - passwd table
 - adding contents to /etc/passwd file, 416
 - changing password information, 20
 - columns, 415
 - credential-related information, 85, 324
 - described, 45, 414
 - shadow column, 415
 - nistbladm command and, 149 to 152, 164 to 165
 - User ID 0, 332
 - passwd.byname NIS map, 11
 - passwd.byuid NIS map, 11
 - password expired message, 139 to 140
 - password will expire message, 140, 162

-
- passwords, 144 to 172
 - See also* security
 - aging, 158 to 168
 - aging vs. expiration, 164
 - calculating dates, 152 to 153
 - date of last change, 150
 - described, 158 to 159
 - expiration date for password
 - privilege, 151 to 152, 164 to 166
 - forcing users to change passwords, 159
 - maximum days before changing, 150 to 151, 160, 170
 - maximum days between logins, 151, 166 to 167
 - MAXWEEKS setting, 170
 - minimum days before changing, 150, 161, 170
 - MINWEEKS setting, 170
 - nobody class and, 147
 - operations affected, 159
 - PASSLENGTH setting, 169, 171
 - passwd file defaults, 168 to 171
 - setting criteria for multiple users, 168
 - turning off aging, 163
 - turning off privilege expiration, 166
 - warning of required change, 151, 162 to 163, 170
 - WARNWEEKS setting, 170
 - world class and, 147
 - bad choices, 143
 - changing, 156 to 157
 - another's password, 156
 - choosing passwords, 142 to 143
 - failures, 142, 171
 - forcing users to change passwords, 159
 - maximum unsuccessful attempts, 171
 - root password, 141, 157
 - your password, 140 to 142, 156
 - commands, 69
 - displaying information, 154 to 155
 - dummy password for creating credential information, 93 to 95
 - failure limits, 171 to 172
 - locking passwords, 157
 - logging in
 - See also* logging in
 - maximum login time period, 172
 - process, 138
 - troubleshooting, 139 to 140
 - minimum length, 171
 - nispaswd command, 144
 - nistbladm command, 148 to 153
 - calculating number of days, 152 to 153
 - nsswitch.conf file
 - requirements, 144
 - passwd command, 145 to 148
 - passwd table, 414 to 416
 - related commands, 154
 - requirements, 142 to 143
 - Solaris access, 54
 - specifying criteria and defaults, 168 to 172
 - troubleshooting
 - forgotten password, 339
 - "login incorrect" message, 139, 157, 166, 321
 - NIS+ password and login password in /etc/passwd file, 330
 - password change failures, 142
 - password locked, expired, or terminated, 139 to 140, 322
 - "password [problems]" messages, 321
 - root password change causes problem, 332
 - Secure RPC password different from login password, 76, 82 to 83, 99 to 100, 330 to 331

Sorry: less than *N* days
 since the last change
 message, 141, 161

Too many failures - try
 later message, 139, 142,
 172

Too many tries; try again
 later message, 139, 142

user cannot change
 password, 340

user cannot log in after password
 change, 313 to 314, 340

You may not change this
 password message, 141

Your password has been
 expired for too long
 message, 160

Your password has expired
 message, 139 to 140

Your password will expire
 message, 140, 162

unlocking passwords, 158

using, 138 to 143

yppasswd command, 145

paths

- NIS_PATH variable, 43 to 44, 333
- search paths for NIS+ tables, 48 to 50
- table paths and performance, 44, 333

performance problems, 332 to 336

- checkpointing in progress, 333
- large NIS+ database logs (slow
 startup), 334
- nis_cachemgr missing, 335
- NIS_PATH variable too complex, 44,
 333
- niscat causes Server busy
 message, 336
- recursive groups, 334
- startup slow, 334, 335
- symptoms, 332 to 333
- table paths, 333
- too many replicas, 334

periods, *See* dots (.)

“permission denied” message, 322

- during login attempt, 140
- /etc/.rootkey file preexisting, 331
- nobody authorization class, 82 to 83
- user cannot login after password
 change, 313
- user cannot remote log in to remote
 domain, 340
- user login same as machine
 name, 320

permissions

- See also* access rights; authorization;
 security
- changing, 283
- checking, 282 to 283, 319
- network wide groups, 411
- passwd command and, 147 to 148

Solaris

- file and directory matrix, 55
- NIS+ objects matrix, 56

troubleshooting, 318 to 321

- credentials missing or
 corrupted, 319, 327 to 328
- “insufficient permission”
 message, 318, 321, 331
- no permission, 319
- “no permission” messages
 (FNS), 344
- “permission denied”
 message, 313, 320, 322,
 331
- server running at security level
 0, 319
- symptoms, 318
- user login same as machine
 name, 319 to 320

pinging replica servers, 199 to 200

pipe symbol (|) as regular expression
 symbol, 220

plus sign (+)

- access rights operator, 121, 122
- nsswitch.conf compatibility with
 +/- syntax, 235, 242 to 243
- regular expression symbol, 220

Possible loop detected in
 namespace message, 310

pound sign (#)

- nsswitch.conf file comments, 238
- shell prompts, xxxii
- prefix of Secure RPC netnames, 80
- principals (NIS+)
 - authorization classes, 63 to 67, 110
 - authorization process
 - login phase, 75, 76 to 77
 - preparation phase, 75, 76
 - request phase, 76, 77 to 79
 - changing keys
 - chkey command, 100 to 102
 - nonroot server keys, 105
 - root keys from another machine, 104
 - root keys from root, 102
 - root replica keys from the replica, 104 to 105
 - creating credential information, 91 to 96
 - dummy password example, 93 to 95
 - other domain example, 95
 - overview, 92
 - user principals example, 93
 - workstation example, 96
 - defined, 30
 - described, 58
 - fully qualified names, 42
 - naming syntax, 91
 - Secure RPC netnames vs., 42
 - testing for group membership, 182
- printer context type
 - administration
 - using files, 302
 - using NIS, 302
 - using NIS+, 303 to 304
 - creation, 264
- printers
 - context administration
 - using files, 302
 - using NIS, 302
 - using NIS+, 303 to 304
 - context creation, 264
- printing references used for binding, 273, 274
- private keys
 - See also* keys; public keys
 - authentication process, 77 to 79
 - changing
 - chkey command, 100 to 102
 - nonroot server keys, 105
 - root keys from another machine, 104
 - root keys from root, 102
 - root replica keys from the replica, 104 to 105
 - commands, 69
 - creating DES credentials, 81 to 82
 - creation by nisaddcred, 89
 - logging out and security, 76 to 77
 - NIS map, 11
 - passwd command and, 148
 - re-encrypting existing key, 100, 101
 - Secure RPC password different from
 - login password, 76, 82 to 83, 99 to 100
 - where information is stored, 84
 - problems, *See* troubleshooting (FNS); troubleshooting (NIS+)
 - processes
 - checking size, 337
 - insufficient, 338
 - prompts, default, xxxii
 - protocols table
 - columns, 416
 - described, 46, 416
 - protocols, NIS maps, 11
 - protocols.byname NIS map, 11
 - protocols.bynumber NIS map, 11
 - public keys
 - See also* keys; private keys
 - authentication process, 77 to 79
 - cached public keys problems, 83 to 84
 - changing, 326 to 327
 - chkey command, 100 to 102
 - nonroot server keys, 105
 - root keys from another machine, 104
 - root keys from root, 102

- root replica keys from the replica, 104 to 105
 - checking directories for public keys, 88
 - commands, 69
 - creating DES credentials, 81 to 82
 - creation by `nisaddcred`, 89
 - generating, 76, 325
 - NIS map, 11
 - propagating directory objects into client files, 326
 - propagating to directory objects, 325
 - updating, 20, 105 to 107, 324 to 327
 - examples, 107
 - `nisupdkeys` command arguments, 106
 - `nisupdkeys` command overview, 105 to 106
 - updating IP addresses, 107
 - where information is stored, 84 to 85, 323
 - `publickey.byname` NIS map, 11
- Q**
- queries (NIS+) hang after changing host name, 336 to 337
 - question mark (?) as regular expression symbol, 220
 - quitting, *See* stopping
 - quotation marks (") in fully qualified names, 42
- R**
- R
 - `nisl` option, 185
 - `nistbladm` option, 215
 - r
 - `fnbind` option, 273
 - `fncheck` option, 279, 280
 - `fncreate` option, 258
 - `fncreate_fs(1M)` option, 294
 - `keylogin` option, 100
 - `nisaddcred` option, 97
 - `nisaddent` option, 226, 227
 - `nisclient` option, 245 to 246
 - `nisdefault` option, 124, 125
 - `nisgrpadm` option, 177, 181
 - `nisinit` option, 195, 196
 - `nistbladm` option, 214
 - `passwd` option, 146
 - r access rights, *See* read access rights
 - RAM, insufficient, 337
 - random (DES) key, 77, 81
 - read access rights
 - See also* access rights described, 67 to 68, 110
 - levels and, 116
 - syntax, 122
 - recursive directory listing, 185
 - recursive group members, 174, 175
 - recursive group nonmembers, 175
 - re-encrypting existing private key, 100, 101
 - references
 - binding composite names to, 273 to 275
 - NIS+ root reference, 285 to 286
 - printing references used for binding, 273, 274
 - regular expressions, 220
 - remote logins
 - `netgroup` table, 411 to 413
 - user cannot remote log in to remote domain, 340
 - remote mounts, `netgroup` table, 411 to 413
 - removing
 - See also* destroy access rights
 - access rights
 - columns, 132
 - objects or entries, 129
 - attributes or values, 277
 - cannot delete `org_dir` or `groups_dir`, 311
 - clearing
 - directory cache, 197
 - keys, 106

- composite names from
 - namespace, 275 to 276
 - credential information, 97 to 98
 - credentials, 88
 - destroying named objects, 276
 - directories, 190 to 191
 - disassociating replicas from
 - directories, 191
 - forcing removal without proper
 - permissions, 191, 192
 - group members, 177, 181
 - groups, 177, 179
 - NIS+, 245 to 250
 - from client machine, 245 to 246
 - from server, 246 to 248
 - NIS+ directories and replicas from
 - namespace, 19
 - NIS+ namespace, 248 to 250
 - NIS+ objects from namespace, 19
 - NIS+ table entries
 - multiple entries, 215
 - single entry, 214
 - nondirectory objects, 191 to 192
- renaming bindings, 275
- replacing attributes or values, 278
- replica servers
 - See also* servers
 - adding to domain, 326
 - adding to existing directory, 187, 189
 - to 190
 - changing keys
 - nonroot servers, 105
 - root replica keys from the
 - replica, 104 to 105
 - defined, 9, 12, 15
 - described, 28
 - disassociating from directories, 191
 - displaying time of last update, 198, 199
 - down or out of service, 309
 - lagging or out of sync, 316
 - performance issues, 334
 - pinging, 199 to 200
 - removing from namespace, 19
 - resynchronization, 30
 - update failure, 341 to 342
 - updating process, 28 to 30
- `replica_update` error messages, 341 to 342
- replicating FNS service, 256
- requirements
 - FNS in NIS+, 254
 - passwords, 142 to 143
- resource problems, 337 to 338
 - insufficient disk space, 310, 338
 - insufficient memory, 337
 - insufficient processes, 338
 - symptoms, 337
- resource requirements, 254
- resynchronization of replica servers, 30
- return switch action option, 235, 236
- `rlogin` command
 - password aging and, 159
 - user cannot remote log in to remote
 - domain, 340
- root
 - changing keys
 - from another machine, 104
 - from root, 102
 - root replica from the replica, 104
 - to 105
 - changing passwords, 141, 157
 - NIS+ namespace vs. UNIX file
 - system, 23
 - reference for NIS+, 285 to 286
 - security issues, 57, 60
 - UID, 62
- root directory, namespace, 24
- root master server
 - defined, 28
 - initializing, 195, 196
- root password
 - change causes problem, 332
 - `passwd` table and, 414
 - Solaris security, 55
- `root_replica_update`: update
 - failed message, 342
- `.rootkey` file

- changing machine to different domain, 329
 - preexisting, 331
 - removing
 - when removing NIS+ from client, 246
 - when removing NIS+ namespace, 248
 - RPC error messages, 350
 - RPC table
 - columns, 416
 - described, 46, 416
 - example, 417
 - rpc.bynumber NIS map, 11
 - rpc.nisd command
 - master rpc.nisd daemon died, 334 to 335
 - multiple parent processes, 311 to 312
 - options, 193
 - B, 193, 194, 312, 337
 - M, 334
 - overview, 192 to 193
 - “server not responding” message, 334
 - slow startup, 334
 - starting the NIS+ daemon, 192 to 194
 - DNS-forwarding NIS-compatible daemon, 193, 194
 - NIS-compatible daemon, 193 to 194
 - rpc.nisd_resolv daemon, rpc.nisd with -B option and, 312, 337
 - rpc.nispasswd daemon, password failure limits, 171 to 172
 - RPCs, NIS map, 11
 - running, *See* starting
- S**
- s
 - fnattr option, 278
 - fnbind option, 273
 - fncheck option, 279
 - fncreate option, 258
 - nisdefault option, 124, 125
 - nismkdir option, 187, 189
 - nismrmdir option, 191
 - nisupdkeys option, 106
 - passwd option, 154 to 155
 - S column type, 210
 - S, rpc.nisd option, 193
 - search criteria for nsswitch.conf file, 235
 - defaults, 236 to 237
 - search paths for NIS+ tables, 48 to 50
 - searching for NIS+ table entries, 19
 - first column 229, 221
 - multiple columns, 222
 - particular column, 222
 - regular expressions, 220
 - secret keys, *See* private keys
 - Secure RPC
 - credential/authorization process, 59
 - netname
 - creation by nisaddcred, 89
 - DES credentials, 80
 - syntax, 90
 - NIS maps, 11
 - NIS+ principal names vs., 42
 - password different from login password, 76, 330 to 331
 - Solaris security, 55
 - You *name* do not have Secure RPC credentials error message, 319
 - security, 53 to 69
 - See also* access rights; credentials; passwords; permissions
 - administrative rights and, 68
 - authentication
 - credential/authentication process, 74 to 79
 - defined, 12, 17
 - DES credentials, 60 to 61
 - described, 56
 - process, 56 to 57
 - Solaris Secure RPC gate, 55
 - authorization
 - defined, 12, 17

- described, 56, 63
- NIS+ authorization classes, 63 to 67, 110
- passwd command
 - requirements, 148
- process, 57
- default values, setting, 125 to 127
- described, 12, 16 to 17
- levels
 - described, 58 to 59
 - password commands and, 59
 - servers running at level 0, 319
 - setting for NIS+ daemon, 193
- NIS vs. NIS+, 13
- NIS+
 - access rights, 67 to 68
 - administrator, 68
 - authorization classes, 63 to 67, 110
 - commands, 69
 - credentials and authentication, 59 to 62
 - NIS vs. NIS+, 13
 - overview, 56 to 58
 - password commands, 59
 - principals, 58
 - security levels, 58 to 59
- NIS-compatibility mode issues, 18, 55
- Solaris, 53 to 56
- specifying nondefault security values, 127 to 128
- troubleshooting, 321 to 332
 - cached public keys problems, 83 to 84
 - credential information
 - outdated, 322 to 327
 - credentials corrupted, 327 to 328
 - domain change for machine, 329
 - domain name changed, 329
 - /etc/.rootkey file
 - preexisting, 331
 - key serv failure, 328
 - “login incorrect” message, 139, 157, 166, 321
 - machine previously was NIS+ client, 329
 - NIS+ password and login password in
 - /etc/passwd file, 330
 - no entry in cred table, 329
 - password locked, expired, or terminated, 322
 - root password change causes problem, 332
 - Secure RPC password different from login password, 76, 82 to 83, 99 to 100, 330 to 331
 - server running at security level 0, 319
 - symptoms, 321
- Security exception on LOCAL system message, 318, 320
- sendmail program, mail aliases table, 411
- Server busy. Try again message, 30, 336
- “server not responding” message, 334
- servers
 - access rights, granting to tables, 119 to 120
 - changing keys
 - nonroot servers, 105
 - root keys from another machine, 104
 - root keys from root, 102
 - root replica keys from the replica, 104 to 105
 - clients as, 36 to 37
 - clients vs., 27
 - client-server computing, 4
 - disassociating from directories, 190 to 191
 - displaying time of last update, 198, 199
 - /etc/.rootkey file preexisting, 331
 - home domain, 36 to 37
 - initializing NIS+ servers, 19
 - master servers
 - defined, 9, 12

- listing directory contents
 - from, 185
- listing table entries from, 216, 221
- NIS, 9
- NIS+, 12, 15
- replicas and, 28
- root master server, 28, 195, 196
- NFS file servers, 292 to 293
- NIS map, 11
- NIS+
 - connection to domains, 27
 - ctx_dir directory, 256
 - directories, 22, 255
 - overview, 27 to 28
 - requirements for FNS, 254
- NIS-compatibility mode, 18
- removing NIS+, 246 to 248
- replicas
 - adding to domain, 326
 - adding to existing directory, 187, 189 to 190
 - changing root replica keys from the replica, 104 to 105
 - defined, 9, 12, 15
 - described, 28
 - disassociating from directories, 191
 - down or out of service, 309
 - lagging or out of sync, 316
 - performance issues, 334
 - pinging, 199 to 200
 - removing from namespace, 19
 - resynchronization, 30
 - update failure, 341 to 342
 - updating process, 28 to 30
- security level 0, 319
- Server busy. Try again message, 336
- “server not responding” message, 334
- slow performance after NIS+ installation, 335
- update implementation, 28 to 30
- updating NIS+ tables, 52
- service context type, 263
- services
 - context creation, 263
 - context ownership, 263
- services table, 46, 417
- services.byname NIS map, 11
- setenv command, 126 to 127
- setting up FNS
 - FNS namespace setup, 255 to 256
 - NIS+ service setup, 254 to 255
 - replicating FNS service, 256
 - resource requirements, 254
- severity threshold for error reporting, 349
- shared directory cache, *See* directory cache
- shells
 - changing NIS_DEFAULTS values, 126 to 127
 - default prompts, xxxii
 - netgroup table, 411 to 413
- simple names, 37
- single host context, creating, 260
- single user context, creating, 262
- site context type, 265
- sites
 - context creation, 265
 - context ownership, 265
- slash (/)
 - double slashes in organization names, 283
 - trailing, in organization names, 284
 - in UNIX filenames, 23
- slave servers, *See* replica servers
- slow performance, *See* performance problems
- Solaris, 17, 18
 - group table, 409 to 410
 - security, 53 to 56
 - dialup gate, 54
 - file and directory permissions matrix, 55
 - login gate, 55
 - NIS+ objects permissions matrix, 56

-
- root gate, 55
 - schema of gates and filters, 54
 - Secure RPC gate, 55
 - Solstice System Management Tools,
 - creating credential information, 86
 - Sorry: less than *N* days since the last change message, 141, 161
 - spaces, column separators for NIS+ tables, 404
 - starting
 - NIS+ cache manager, 18, 197
 - NIS+ daemon, 192 to 194
 - DNS-forwarding NIS-compatible daemon, 193, 194
 - NIS-compatible daemon, 193 to 194
 - stopping
 - cache manager, 197
 - NIS+ daemon, 194
 - storage requirements for FNS on NIS+, 254
 - storing credential information, 84 to 85, 322 to 324
 - su command and security, 57
 - subnetting
 - name service advantages, 4 to 8
 - network masks table, 413
 - suborganizations
 - fnlist does not list, 345
 - listing using nisls, 345
 - SUCCESS switch status message, 235, 236
 - default response, 237
 - superusers
 - See also root password
 - default prompts, xxxii
 - switch, the, See name service switch
 - symbolic links
 - creating, 19, 223
 - following by NIS+ commands, 222
 - syslog.conf file, severity threshold for error reporting, 349
 - system hang problems, 332 to 337
 - master rpc.nisd daemon died, 334 to 335
 - NIS+ queries hang after changing host name, 336 to 337
 - NIS_PATH variable too complex, 333
 - symptoms, 332 to 333
 - system resource problems, 337 to 338
 - insufficient disk space, 338
 - insufficient memory, 337
 - insufficient processes, 338
 - symptoms, 337
 - system tables (NIS+)
 - See also NIS+ tables
 - creating, 50 to 51
 - described, 16, 45 to 46
- ## T
- t
 - fncheck option, 279
 - fncreate option, 258, 345
 - nisaddent option, 229, 231
 - nisdefault option, 124, 203
 - nisgrpadm option, 177, 182
 - nislog option, 201
 - table level
 - access rights
 - create rights, 116
 - described, 113
 - destroy rights, 117
 - example, 114 to 115
 - modify rights, 117
 - read rights, 116
 - syntax, 123
 - types and objects acted upon, 116
 - authorization classes and, 66
 - tables (/etc), nsswitch.conf
 - specification of sources, 404
 - tables (NIS+), 45 to 52, 403 to 418
 - See also column level; entry level; nistbladm command; table level; *specific tables*
 - access rights, 113 to 117

- how servers grant rights, 119 to 120
- adding entries, 211 to 213
- adding information from `/etc` files or NIS maps, 18
- administration, 207 to 231
- alternate sources of information, 233 to 234
- changing entries, 213 to 214
- changing time-to-live for entries, 203, 205
- clients and, 31
- column requirements, 209
- column separators, 404
- column types, 210
- creating
 - unpopulated tables, 19, 50 to 51
 - using `fncreate` command, 257
 - using `nistbladm` command, 209 to 210
- creating links, 223
- displaying contents, 18
- dumping table information to a file, 231
- `/etc` files corresponding to, 404
- expanding directories into NIS+ domain, 225
- expanding directories into NIS-compatible domain, 225
- fully qualified names, 40 to 41
- indexed names, 41
- input file format requirements, 404
- listing object properties, 217 to 218
- listing table contents, 217
- loading information, 226 to 231
 - loading from NIS maps, 227, 229 to 231
 - loading from text files, 227, 228 to 229
 - methods of populating tables, 51 to 52
 - replacing, merging, or appending, 226
- NIS maps and matching tables, 229 to 230
- `nisaddent` command, 226 to 231
- `niscat` command, 216 to 218
- `nisgrep` command, 219 to 222
- `nisl` command, 222 to 223
- `nismatch` command, 219 to 222
- `nissetup` command, 224 to 225
- `nistbladm` command, 208 to 216
- `nsswitch.conf` specification of sources, 404
- paths, 333
- preconfigured tables, 16, 45 to 46
- removing, 211
- removing entries
 - multiple entries, 215
 - single entry, 214
- searching for entries, 19
 - first column 229, 221
 - multiple columns, 222
 - particular column, 222
 - regular expressions, 220
- setting up, 50 to 52
- structure, 45 to 50
 - columns and entries, 47 to 48
 - search paths, 48 to 50
- system tables, 16, 45 to 46
 - creating, 50 to 51
- updating process, 52

tabs as column separators for NIS+ tables, 404

`telnet` command and password aging, 159

template files for `nsswitch.conf`

- default template, 241
- described, 233 to 234
- examples, 240 to 241
- overview, 239

terminated password, 322

terminating. *See* stopping

time stamps

- authentication process, 77 to 79, 81 to 82
- transaction log, 29

time-to-live (TTL)

- changing field values, 19
- changing for objects, 204 to 205
- changing for table entries, 203, 205

- described, 33, 203 to 204
- displaying values, 203
- timezone table, 46, 418
- /tmp files and disk space problems
 - and, 338
- Too many failures - try later message
 - during login, 139
 - during password change, 142, 172
- Too many tries; try again later message
 - during login, 139
 - during password change, 142
- trans.log file, caution against
 - renaming, 22
- transaction logs (NIS+)
 - cannot truncate, 310
 - corrupted, 311 to 312
 - described, 29
 - displaying contents, 201 to 202
 - replica updating process, 28 to 30
- troubleshooting (FNS), 284, 343 to 348
 - See also* error messages
 - contexts
 - cannot be removed by creator, 346
 - checking naming inconsistencies, 279 to 280
 - debugging creation, 258
 - host-related contexts cannot be created, 345
 - initial context cannot be obtained, 343
 - initial context contains nothing, 343
 - user-related contexts cannot be created, 345
 - fndestroy/fnunbind does not return "operation failed", 347 to 348
 - fnlist does not list suborganizations, 345
 - "name in use" messages, 346 to 347
 - "no permission" messages, 344
- troubleshooting (NIS+), 307 to 348
 - See also* error messages
 - miscellaneous problems, 341 to 342
 - checking if NIS+ is running, 341
 - replica update failure, 341 to 342
 - namespace administration
 - problems, 308 to 311
 - cannot add user to group, 309
 - cannot delete org_dir or groups_dir, 311
 - cannot truncate transaction log file, 310
 - checkpoint fails consistently, 309
 - disk space insufficient, 310
 - domain name confusion, 310 to 311
 - illegal object problems, 308 to 309
 - logs grow too large, 309
 - nisinit fails, 309
 - symptoms, 308
 - namespace database problems, 311 to 312
 - multiple rpc.nisd processes, 311 to 312
 - symptoms, 311
 - NIS compatibility problems, 312 to 314
 - nsswitch.conf file fails to perform correctly, 314
 - symptoms, 312 to 313
 - user cannot log in after password change, 313 to 314
 - nsswitch.conf syntax errors, 238
 - object not found problems, 314 to 318
 - automounter cannot be used, 318
 - blanks in name, 317
 - domain levels not correctly specified, 315
 - files missing or corrupt, 316
 - incorrect path, 315
 - object does not exist, 316
 - replica lagging or out-of-sync, 316
 - symptoms, 315

-
- syntax or spelling error, 315
 - ownership and permission
 - problems, 318 to 321
 - credentials missing or
 - corrupted, 319, 327 to 328
 - no permission, 319
 - server running at security level
 - 0, 319
 - symptoms, 318
 - user login same as machine
 - name, 319 to 320
 - security problems, 321 to 332
 - cached public keys problems, 83 to 84
 - credential information
 - outdated, 322 to 327
 - credentials corrupted, 327 to 328
 - domain name change for machine, 329
 - domain name changed, 329
 - /etc/.rootkey file
 - preexisting, 331
 - keyserver failure, 328
 - “login incorrect” message, 139, 157, 166, 321
 - machine previously was NIS+ client, 329
 - NIS+ password and login
 - password in
 - /etc/passwd file, 330
 - no entry in cred table, 329
 - password locked, expired, or terminated, 322
 - root password change causes
 - problem, 332
 - Secure RPC password different
 - from login password, 76, 82 to 83, 99 to 100, 330 to 331
 - symptoms, 321
 - slow performance problems, 332 to 336
 - checkpointing in progress, 333
 - large NIS+ database logs (slow startup), 334
 - nis_cachemgr missing, 335
 - NIS_PATH variable too
 - complex, 44, 333
 - niscat causes Server busy
 - message, 336
 - recursive groups, 334
 - startup slow, 334, 335
 - symptoms, 332 to 333
 - table paths, 333
 - too many replicas, 334
 - system hang problems, 332 to 337
 - master rpc.nisd daemon
 - died, 334 to 335
 - NIS+ queries hang after changing
 - host name, 336 to 337
 - NIS_PATH variable too
 - complex, 333
 - symptoms, 332 to 333
 - system resource problems, 337 to 338
 - insufficient disk space, 338
 - insufficient memory, 337
 - insufficient processes, 338
 - symptoms, 337
 - user problems, 338 to 341
 - cannot change password, 340
 - cannot log in, 339
 - cannot log in using new
 - password, 340
 - cannot remote log in to remote
 - domain, 340
 - symptoms, 339
 - truncated transaction log file, 310
 - TRYAGAIN switch status message, 236
 - default response, 237
 - TTL, *See* time-to-live (TTL)
 - typographic conventions in this
 - manual, xxxii
- U**
- U
 - fnattr option, 278
 - fnbind option, 273
 - u
 - fncheck option, 279, 280
 - nisping option, 198, 199

-
- nistbladm option
 - adding access rights to existing table column, 131 to 132
 - described, 130
 - removing access rights to table column, 132
 - UID=0, 62
 - “unable to find” message, 315
 - “unable to fork” message, 338
 - “unable to make request” message, 318
 - “unable to stat” message, 315, 318
 - UNAVAIL switch status message, 236
 - default response, 237
 - UNIX groups, group table, 409 to 410
 - “unknown user” message, 313
 - unlocking passwords, 158
 - unsetenv command, 127
 - updating
 - credential information, 322 to 324
 - using nisaddcred, 87 to 88
 - where information is stored, 84 to 85
 - your own credential information, 97
 - directory objects after key change, 104, 105
 - displaying time of last update for server, 198, 199
 - FNS namespace, 279
 - IP addresses, 107
 - NIS+ incremental updates, 13, 15
 - NIS+ tables, 52
 - pinging replica servers, 199 to 200
 - public keys, 20, 105 to 107, 324 to 327
 - examples, 107
 - nisupdkeys command
 - arguments, 106
 - nisupdkeys command
 - overview, 105 to 106
 - updating IP addresses, 107
 - replica update failure, 341 to 342
 - resynchronization of replica servers, 30
 - server propagation of changes, 28 to 30
 - user context type, 262
 - user credentials
 - See also* credentials
 - described, 60
 - types, 62
 - username context type, 261 to 262
 - users
 - context creation
 - all users, 258, 261 to 262
 - single user, 262
 - context ownership, 262
 - credential information creation, 93
 - credential types, 62
 - inactivity maximum, 339
 - troubleshooting NIS+ problems, 338 to 341
 - cannot add to group, 309
 - cannot change password, 340
 - cannot log in, 339
 - cannot log in after password change, 313 to 314, 340
 - cannot remote log in to remote domain, 340
 - forgotten password, 339
 - login same as machine name, 319 to 320
 - symptoms, 339
 - /usr/bin directory, 22
 - /usr/lib directory, 22
 - /usr/lib/nis directory, 22
 - /usr/lib/nis/nissetup
 - command, 19, 51
 - /usr/lib/nis/nisupdkeys command
 - arguments, 106
 - cached public keys problems, 83
 - described, 20, 69
 - examples, 107
 - updating directory objects after key change, 104, 105
 - updating IP addresses, 107
 - updating keys manually, 324 to 327
 - updating public keys, 105 to 107

/usr/sbin directory, 22

V

-v

- fnbind option, 273, 274
- fncreate option, 258
- fncreate_fs(1M) option, 294
- fnlist option, 269, 272, 280
- fnlookup option, 267, 280
- nisaddent option, 227
- nisdefault option, 124, 125
- nislog option, 201

/var/nis directory, 22

/var/nis/data directory, 22, 316

/var/nis/data.dict file, caution
against renaming, 22

/var/nis/NIS_COLDSTART file
changing credentials, 326
creation, 326
credential-related information, 85,
323
initializing clients by, 196
updating keys, 106
See also cold-start file

/var/nis/NIS_SHARED_DIRCACHE file
cached public keys problems, 83 to 84
changing credentials, 326
credential-related information, 84,
323
initializing, 326
starting the cache manager, 18, 196 to
197
See also directory cache; time-to-live
(TTL)

/var/nis/trans.log file, caution
against renaming, 22

verification field of DES credentials, 80

verifying

- checking access control, 282 to 283
- checking context naming
inconsistencies, 279 to 280
- ctx_dir directory creation, 255

viewing, *See* displaying; listing

W

w access rights, 121

warning before password change
required, 151, 162 to 163, 170

WARNWEEKS default for passwords, 170

wildcards in netgroup table input file
format, 412

window for authentication, 77 to 79

workstations

- See also* diskless workstations
- creating credential information, 96
- default timezones table, 418
- Ethernet addresses table, 408 to 409
- initializing workstation clients, 194 to
196

- IP addresses table, 410

- net groups table, 411

- NIS maps, 10

- user groups table, 409 to 410

world authorization class

- access rights, 111

- displaying for objects, 118 to 119

- described, 63, 66

- password-aging constraints and, 147

X

-x

- fnbind option, 273

- passwd option, 159 to 163

X.500 global directory service

- federating NIS+ under, 288 to 289

X/Open Federated Naming

- links

- creating and binding, 273, 274

- displaying binding, 267

- X.500 entry supporting reference
attributes, 288

XFN, *See* X/Open Federated Naming

Y

-Y

- nisaddent option, 231

nissetup option, 225
rpc.nisd option, 193 to 194
-y,nisaddent option, 230
You may not change this password
message, 141
You *name* do not have Secure RPC
credentials error message, 319
Your password has been expired
for too long message, 160
Your password has expired
message, 139 to 140
Your password will expire
message, 140, 162
Your specified repository is not
defined in the nsswitch
file! message, 146
ypcat command
netgroup tables
and, 411
YP-compatibility mode. *See* NIS-
compatibility mode
yppasswd command, 145, 154
ypservers NIS map, 11
ypupdate protocol, 18
ypxfr protocol, 18

Z

Copyright 1995 Sun Microsystems, Inc., 2550 Garcia Avenue, Mountain View, Californie 94043-1100 USA.

Tous droits réservés. Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie et la décompilation. Aucune partie de ce produit ou de sa documentation associée ne peuvent être reproduits sous aucune forme, par quelque moyen que ce soit sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il en a.

Des parties de ce produit pourront être dérivées du système UNIX[®], licencié par UNIX Systems Laboratories Inc., filiale entièrement détenue par Novell, Inc. ainsi que par le système 4.3. de Berkeley, licencié par l'Université de Californie. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

LEGENDE RELATIVE AUX DROITS RESTREINTS : l'utilisation, la duplication ou la divulgation par l'administration américaine sont soumises aux restrictions visées à l'alinéa (c)(1)(ii) de la clause relative aux droits des données techniques et aux logiciels informatiques du DFAR 252.227- 7013 et FAR 52.227-19.

Le produit décrit dans ce manuel peut être protégé par un ou plusieurs brevet(s) américain(s), étranger(s) ou par des demandes en cours d'enregistrement.

MARQUES

Sun, Sun Microsystems, le logo Sun, Solaris, Solstice, AdminTool, AdminSuite sont des marques déposées ou enregistrées par Sun Microsystems, Inc. aux États-Unis et dans certains autres pays. UNIX est une marque enregistrée aux États-Unis et dans d'autres pays, et exclusivement licenciée par X/Open Company Ltd. OPEN LOOK est une marque enregistrée de Novell, Inc., PostScript et Display PostScript sont des marques d'Adobe Systems, Inc.

Toutes les marques SPARC sont des marques déposées ou enregistrées de SPARC International, Inc. aux États-Unis et dans d'autres pays. SPARCcenter, SPARCcluster, SPARCcompiler, SPARCdesign, SPARC811, SPARCengine, SPARCprinter, SPARCserver, SPARCstation, SPARCstorage, SPARCworks, microSPARC, microSPARC II et UltraSPARC sont exclusivement licenciées à Sun Microsystems, Inc. Les produits portant les marques sont basés sur une architecture développée par Sun Microsystems, Inc.

Les utilisateurs d'interfaces graphiques OPEN LOOK[®] et Sun[™] ont été développés par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique, cette licence couvrant aussi les licences de Sun qui mettent en place OPEN LOOK GUIs et qui en outre se conforment aux licences écrites de Sun.

Le système X Window est un produit du X Consortium, Inc.

CETTE PUBLICATION EST FOURNIE "EN L'ÉTAT" SANS GARANTIE D'AUCUNE SORTE, NI EXPRESSE NI IMPLICITE, Y COMPRIS, ET SANS QUE CETTE LISTE NE SOIT LIMITATIVE, DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DES PRODUITS À RÉPONDRE À UNE UTILISATION PARTICULIÈRE OU LE FAIT QU'ILS NE SOIENT PAS CONTREFAISANTS DE PRODUITS DE TIERS.

CETTE PUBLICATION PEUT CONTENIR DES MENTIONS TECHNIQUES ERRONÉES OU DES ERREURS TYPOGRAPHIQUES. DES CHANGEMENTS SONT PÉRIODIQUEMENT APPORTÉS AUX INFORMATIONS CONTENUES AUX PRÉSENTES, CES CHANGEMENTS SERONT INCORPORÉS AUX NOUVELLES ÉDITIONS DE LA PUBLICATION. SUN MICROSYSTEMS INC. PEUT RÉALISER DES AMÉLIORATIONS ET/OU DES CHANGEMENTS DANS LE(S) PRODUIT(S) ET/OU LE(S) PROGRAMME(S) DÉCRITS DANS CETTE PUBLICATION À TOUTS MOMENTS.

